Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava 4

Tímový projekt
RoboCup 3D

Dokumentácia k inžinierskemu dielu

Študijný odbor: Softvérové inžinierstvo, Informačné systémy
Predmet: Tímový projekt
Akademický rok: 2012/2013
Téma: RoboCup 3D
Číslo tímu: 15
Názov tímu: A55 Kickers
Vedúci tímu: Ing. Marián Lekavý, PhD.
Kontakt: tptim15@googlegroups.com

Bc. Matej Červeňák
Bc. Jaroslav Grega
Bc. Martin Gregor
Bc. Gábor Nagy
Bc. Matúš Ondrejkovič
Bc. Filip Sucháč

Obsah

História zmien

Tabuľka 3.1: História zmien

Zmena	Vykonaná zmena	Dátum zmeny	Vykonal	Platné od verzie
1.	Vytvorená kostra dokumentu	11.11.2012	M. Červeňák	1.1
2.	Pridané časti: Úvod, Plán projektu na zimný semester, Šprinty, Analýza tímu Androids, Analýza tímu rUNSWift, Projekt od konkurenčného tímu, Vytvorenie nového pohybu, Analyzovanie RoboCup wikipedie, Návrh celkovej práce počas projektu, Tvorba dokumentácie	13.11.2012	M. Červeňák	1.1
3.	Pridané časti: Analýza tímu High 5, Analýza tímu beeStambul, Manažérsky softvér	13.11.2012	G. Nagy	1.1
4.	Pridané časti: Analýza tímu Austin Villa, Webová stránka, Server na RoboCup, Spustenie hráča na server, Git, Analýza zdrojových kódov z minulého roka	13.11.2012	M. Gregor	1.1
5.	Pridané časti: Analýza tímu TÍM 17 ŽIJE, Analýza tímu Nexus3D, Analýza robota ASIMO, Návod na písanie kódu	13.11.2012	F. Sucháč	1.1
6.	Pridané časti: Analýza tímu magmaOffenburg, Analýza pohybov	13.11.2012	J. Grega	1.1
7.	Pridané časti: Analýza tímu Karachi Koalas, Analýza pohybov	13.11.2012	M. Ondrejkovič	1.1

Zoznam skratiek

Tabuľka 1.2: Zoznam skratiek

Skratka	Vysvetlenie
pp	Používateľský príbeh
tp	Technický príbeh

1 Úvod

Tento dokument predstavuje dokumentáciu k inžinierskemu dielu tímového projektu *RoboCup – tretí rozmer*, ktorý je vypracovávaný členmi tímu 15 s názvom *A55 Kickers*. Riešenie tímového projektu je obsahom kurzu Tímový projekt na FIIT STU v akademickom roku 2012/2013.

Dokument vytvoril tím 15 a spolu s ostatnými dokumentmi patrí medzi ich internú dokumentáciu. Slúži členom samotného tímu, ako i vedúcemu celého projektu. Takisto ho môžu využiť i ostatní čitatelia ako inšpiráciu pri riešení podobných projektov. Dokument obsahuje tri kapitoly.

V druhej kapitole sa nachádza plán projektu na zimný semester. Plán je rozdelený na jednotlivé týždne resp. šprinty, ku ktorým sa viažu naplánované úlohy.

V tretej kapitole sú spísané jednotlivé šprinty a ich náplne. Ku každému šprintu sú zobrazené používateľské príbehy a úlohy, ktoré sa k príbehom viažu. Obsahom tejto kapitoly sú tiež detailné popisy príbehov a úloh.

2 Plán projektu na zimný semester

V tíme vývoj postupuje metódou Scrum v dvojtýždňových intervaloch, kde na konci každého šprintu zhodnotíme čo sme urobili.

Prvé tri týždne slúžili na výber témy projektu a dohodnutie štruktúry tímu.

V zimnom semestri sú naplánované 4 šprinty. Prvé dva budú zamerané na analýzu problémovej časti a následne dva na návrh a prvotnú implementáciu prototypu.

Šprint/Týždeň	Úlohy	Obdobie
1. týždeň	Výber témy tímového projektu Vypracovanie ponúk	24.9.2012 - 30.9.2012
2. týždeň	Pridelenie tém	1.10.2012 - 7.10.2012
1. šprint	Analýza minuloročných tímov Analýza zahraničných tímov Analýza fyzikálnych modelov robotov Analýza wikipedie RoboCupu Manažérsky softvér Web stránka tímu Inštalácia simulačného prostredia Prvé pohyby robotov	10.10.2012 - 31.10.2012
2. šprint	Návod na používanie Git-u Návod na písanie kódu Analýza zdrojových kódov z minulého roka Analýza chôdze hráča Zlúčenie zdrojových kódov tímu 5 a 17 Návrh plánu projektu Vytvorenie dokumentácie	31.10.2012 - 14.10.2012
3. šprint	Vylepšenie low-lewel pohybov (chôdza, kráčanie za loptou, úkroky, kopnutie do lopty...)	14.10.2012 - 28.10.2012
4. šprint	Vytvorenie prototypu Doplnenie chýbajúcej dokumentácie	28.10.2012 - 12.20.2012

3 Šprinty

V tejto časti dokumentu sú opísané jednotlivé šprinty a k nim zodpovedajúce používateľské príbehy, technické príbehy a úlohy.

Keďže v prvých dvoch šprintoch bola našou úlohou analýza stávajúceho systému vytvoreného v predchádzajúcom roku, nebol vytvorený produkt backlog. Ten sa začal napĺňať až počas 2. šprintu. Vykonávanie jednotlivých užívateľských a technických príbehov, obsiahnutých v backlogu, bolo náplňou až ďalších šprintov. No z dôvodu prehľadnejšieho zápisu uvažujeme v prvých dvoch šprintoch jednotlivé úlohy daných šprintov ako používateľské resp. technické príbehy. Číslovanie jednotlivých úloh zostáva zachované.

3.1 Šprint 1 – „alfa“

Tabuľka 3.1: Príbehy v šprint backlogu

ID pp	Ako	Chcem	Aby bolo možné
1.1	Člen tímu	Vlastniť webovú stránku	Zhromažďovať a zverejňovať všetky informácie o projekte na jednom mieste
1.2	Člen tímu	Funkčný server na RoboCup	Spúšťanie hráča a testovacieho frameworku
1.3	Člen tímu	Získať projekt od konkurenčného tímu z minulého roka	Využiť funkcionalitu obsiahnutú v ich projekte
1.4	Člen tímu	Spustiť hráča na server	Vyvíjať a testovať novú funkcionalitu
1.5	Člen tímu	Vytvoriť nový pohyb hráča	Ľahšie pochopiť nasledujúci vývoj
1.6	Člen tímu	Analyzovať projekty z minulých rokov	Pochopiť a zistiť, čo je alebo nie je implementované
1.7	Člen tímu	Analyzovať wikipediu RoboCupu	Pochopiť a zistiť, čo je alebo nie je implementované
1.8	Člen tímu	Analyzovať zahraničné úspešné tímy	Zistiť možné prístupy a následne ich prípadne navrhnúť a zapracovať
1.9	Člen tímu	Analyzovať fyzikálnych modelov robotov	Zistiť možné prístupy a následne ich prípadne navrhnúť a zapracovať
1.10	Člen tímu	Server na Git	Verziovať zdrojový kód
1.11	Člen tímu	Mať nainštalovaného klienta na Git	Verziovať zdrojový kód
1.12	Člen tímu	Mať manažérsky softvér	Lepšie manažovať riadenie tímu

Tabuľka 3.2: Detailný opis úloh

ID pp	Zodpovedný pp	ID úlohy	Úloha	Zodpovedný
1.1	-	1.1	Rozbehať webovú stránku	M. Gregor
1.2	-	1.2	Rozbehať server na RoboCup.	M. Gregor
1.3	-	1.3	Získať projekt od konkurenčného tímu z posledného roka.	M. Červeňák
1.4	-	1.4	Spustiť hráča na server	M. Gregor

1.5	-	1.5	Vytvoriť pohyb alebo upraviť existujúci pohyb hráča v editore.	všetci
1.6	-	1.6	Študovať dokumentáciu a projekty z minulých rokov.	všetci
1.7	-	1.7	Študovať wiki k RoboCup-u.	všetci
1.8	-	1.8	Študovať zahraničné tímy a zápasy.	všetci
1.9	-	1.9	Analyzovať robotov a prečítať si o metódach učenia, stabilizácie...	M. Ondrejkovič J. Grega F. Sucháč
1.10	-	1.10	Rozbehať Git	Martin Gregor
1.11	-	1.11	Nainštalovať Git u seba.	všetci
1.12	-	1.12	Rozbehať manažérsky softvér	Gábor Nagy

3.1.1 Webová stránka (pp 1.1)

3.1.1.1 Špecifikácia

Webové sídlo projektu musí byť dostupné na webe cez adresu http://labss2.fiit.stuba.sk/TeamProject/2012/teamNNSS/, kde `NN` je číslo tímu (pre IS a SI 01 až 15, pre PKSS 01 až 06) a `SS` je skratka študijného programu/programov (`pss` alebo `is-si`). Všetky súčasti webového sídla sa musia nachádzať na našom serveri (t.j. nie presmerovať stránku niekde inde). Stránky musia byť na serveri umiestnené v DocumentRoot, takže sa zobrazia pri prístupe na tento server. Webové sídlo má informovať o postupe prác tímu a treba ho pravidelne (týždenne) aktualizovať. Na konci projektu náš tím vytvorí a odovzdá statický obraz webového sídla a takto ho bude prezentovať na elektronickom médiu a aj na webe.

Minimálny obsah stránky

- názov témy
- riešitelia a základné informácie o nich (napr. z ponuky)
- plán projektu (na semester)
- aktuálny stav plnenia plánu (t.j., úlohy, ktoré vyplynuli zo stretnutí, ich plnenie a vzťah k plánu)
- roly jednotlivých členov tímu (aj dočasné)
- odkazy na doteraz vypracovanú dokumentáciu vrátane záznamov zo stretnutí (záznamy zo stretnutí budú priamo čitateľné na webe, t.j. najlepšie vo formáte pdf, prípadne HTML)
- všetko zaujímavé v súvislosti s projektom a postupom prác na projekte, napr. odkazy na iné zdroje súvisiace s témou projektu

3.1.1.2 Analýza

Vhodným prostredím na rýchle a udržiavateľné informovanie o stave projektu je systém na správu obsahu ako napríklad wordpres, drupal a iné. CMS (Content Management System) systémy ponúkajú množstvo modulov na uľahčenie a podporu tvorby rôznych typov obsahu ako sú texty, zoznamy, triedenie podľa taxonómie. Sú to serverové aplikácie, ktoré pre správnu funkčnosť vyžadujú na serveri nainštalovaný PHP procesor a databázu ako napríklad PostgreSQL alebo MySQL.

Školský server pristupuje k našim serverom cez reverznú proxy. CMS systémy majú možnosť v konfigurácii nastaviť prístup prostredníctvom reverznej proxy.

3.1.1.3 Návrh

Vytvorili sme stránku na CMS systéme Drupal. Stránka má funkcionalitu pridávania obsahu aktualít, kde sa dá nastaviť dátum, ku ktorému sa aktualita viaže. Ďalej na stránke sa dá vytvoriť typ obsahu súbor, ktorý sa dá zaradiť do kategórie pomocou priradenia výrazu z taxonomického slovníka. Dá sa vytvoriť aj obsah typu webový dokument, kde môžeme vytvárať webové tabuľky, číslované/nečíslované zoznamy, pridávať odkazy a aj obyčajný alebo rôzne formátovaný text. V stránke existuje špeciálny výpis zoznamu aktualít zoradených podľa dátumu, ku ktorému sa viažu a zoznam súborov utriedených do kategórií z taxonomického slovníka. Z obsahových častí obsahuje stránka ešte úvodnú stránku, výpis informácií o členoch tímu a tiež stránku s užitočnými odkazmi použitými počas vývoja produktu. Prístup k stránkam je v prehľadnom menu. Pridávanie obsahu môže len prihlásený používateľ, ktorého overenie a bezpečnosť pri práci zabezpečuje CMS systém Drupal.

3.1.1.4 Implementácia

CMS systém Drupal 7 sme postavili na technológii hypertextového preprocesora PHP 5.3 a databázového servera PostgreSQL 9.1. Všetky technologické komponenty boli nainštalované na webovom serveri Apache 2.2 na operačnom systéme Fedora 17. Aby správne Drupal fungoval potrebovali sme pridať do súboru /sites/default/settings.php nastavenie na akceptovanie cookies súborov cez reverzný proxy server. Do CMS systému sme nainštalovali moduly Views, Taxonomy, Autopath, DatePicker, CKEditor, IMCE. Inštalácia modulov prebehla krokmi:

1. Stiahnutie požadovaných modulov z Drupal portálu.
2. Rozbalenie zazipovaných súborov a ich nakopírovanie do priečinku od koreňa webu /sites/all/modules/
3. Vyčistenie Cache pamäte Drupalu a povolenie nakopírovaných modulov cez administračné rozhranie drupalu.
4. Konfigurácia modulov

Následne sme v systéme vytvorili typy obsahu aktuality s poliami datum, title, body a typ obsahu Dokumenty s poliami title, súbor, kategória. Typ obsahu stránka (webový dokument) je defaultný typ obsahu v CMS systéme Drupal. Vytvorili sme slovník Typy dokumentov, ktorý obsahuje slová (kategórie) Technická dokumentácia, Zápisy stretnutí, Šablóny, Metodiky, Produkt na stiahnutie, Ostatné. Pole kategória v type obsahu Dokumenty berie kategórie z vytvoreného slovníka Typy dokumentov.

3.1.2 Server na RoboCup (pp 1.2)

3.1.2.1 Analýza

Server na RoboCup je Simspark server. Defaultná verzia na stiahnutie je určená pre 32 bitové operačné systém. Pre operačné systémy s 64 bitovou architektúrou je odporúčané si simspark server skompilovať na danom operačnom systéme. Návod je kompiláciu je dostupný Wiki stránke projektu RoboCup. Simspark server vyžaduje nainštalované knižnice ako ruby 1.9 a vyššie, openGL

a iné, ktorých zoznam nájdete v manuáli inštalácie Simspark servera. Simspark server ponúka tri nástroje a to rcssserver, rcssmonitor a rcssagent. Nástroj rcssserver spúšťa server prostredia simulovaného robotického futbalu. Nástroj rcssmonitor graficky vizualizuje svet robotického futbalu. A nakoniec rcssagent ponúka testovacieho agenta, s pustením ktorého si skontrolujeme funkčnosť servera.

3.1.2.2 Návrh

Pri inštalácii na 32 bitovej architektúre operačného systému stačí stiahnuť skompilovaný inštalačný súbor a otestovať spustenie servera a monitora simulovaného robotického futbalu Simspark. Ak nastane chyba a bude obsahovať infomáciu o chýbajúcej knižnici, tak knižnicu doinštalujeme. Na 64 bitových architektúrach treba stiahnuť zdrojové súbory Simsparku a skompilovať ich na danom stroji.

3.1.2.3 Implementácia

Všetci členovia tímu používajú 32 bitové architektúry operačných systémov, čiže pri inštalácii len stiahli skompilované súbory simsparku a server robotického futbalu naištalovali. Z chýbajúcich knižníc hráčom chýbala iba ruby knižnica, ktorú si doinštalovali. Server úspešne spustil každý člen tímu.

3.1.3 Projekt od konkurenčného tímu (pp 1.3)

Keďže sme začali pracovať na projekte, ktorý na škole pôsobí už istý čas, pokračujeme vo vývoji z minulého roka. V minulom roku boli na predmete Tímový projekt dva projekty RoboCup. Teda dva tímy nezávisle od seba vyvíjali rovnaký systém. My sme začali pracovať na jednom z nich. Druhý sme ale k dispozícii nemali, preto bolo potrebné ho získať od minuloročného projektu, aby sme mohli pracovať na zlúčení funkcionality oboch hráčov.

3.1.4 Spustenie hráča na server (pp 1.4)

3.1.4.1 Analýza

Zámerom úlohy bolo spustenie hocijakého hráča na Simspark severi a tým aj otestovanie fuknčnosti servera Simspark. Možnosti ako spustiť hráča sú:

1. Spustiť hráča z nástroja rcssagent
2. Spustiť hráča zo školského projektu Jim
3. Spustiť hráča zo školského projektu TestFramework

3.1.4.2 Návrh

Najrýchlejším riešením je spustenie hráča pomocou nástroja rcssagent. Ale tiež bolo potrebé aj vyskúšať fuknčnosť školských projektov.

3.1.4.3 Implementácia

Každý člen tímu na bežiacom serveri a monitore simulovaného robotického futbalu spustil hráča zo školského projektu Jim a z nástroja rcssagent.

3.1.5 Vytvorenie nového pohybu (pp 1.5)

3.1.5.1 Analýza

Zámerom úlohy bolo vytvorenie nového pohybu alebo upravenie existujúceho pohybu v editore pohybov. Editor pohybov má vypracovaný návod na použitie.

3.1.5.2 Návrh

Najlepším spôsobom ako vytvoriť pohyb je spustenie editoru pohybov a postupovať pomocou návodu na použitie.

3.1.5.3 Implementácia

Každý člen tímu vytvoril nový pohyb hráča, ktorý si overil na bežiacom serveri a monitore simulovaného robotického futbalu.

3.1.6 Analyzovanie dokumentácie z minulých rokov (pp 1.6)

3.1.6.1 Analýza tímu Androids

3.1.6.1.1 Pohyby agenta

Pohyb agenta pozostáva z fáz. Medzi jednoduché pohyby patrí napríklad kopnutie a medzi zložitejšie resp. cyklické napríklad chôdza.

Tím Androids v špecifikácii uvádzal viacero možností, ktoré by radi implementovali. V následujúcej tabuľke sú uvedené všetky. Vypísané sú pohyby, ktoré boli implementované a do kategórie „Návrh do budúcna“ sú zaradené tie, ktoré neboli z časového hľadiska implementované.

Tabuľka 3.3: Pohyby agenta

Pohyb	Implementované	Návrh do budúcna
Vstávanie	vstávanie z brucha vstávanie z chrbta vsávanie zo sedu rozkročmo	vstávanie bez využitia prostredia (posúvaním po ihrisku)
Chôdza	chôdza dopredu chôdza dozadu chôdza do strán (úkroky)	rozdelenie chôdze na rýchlu - nestabilnú a pomalú – stabilnú
Otáčanie	vylepšovanie už použitých pohybov implementovaných v XML vlastné pohyby (90°, 180°, vojenské otočenie)	
Kopanie	vytvorenie XML pre kopanie	

	kopanie dopredu špičkou a stranou porovnanie kopanie celým telom vs. Kopanie kĺbov spodnej končatiny	
Bránenie	bránenie pádom brankára s vystretými rukami do strán bránenie sadnutím na zem a rozkročenie nôh	

XML súbory slúžia na vytváranie zostáv pohybov, ktoré pozostávajú z fáz.

3.1.6.1.2 Vyššia logika

Po analýze agenta JIM predchádzajúceho tímu zistili, že agent nedisponuje vyššími schopnosťami. Tie teda implementovali. Vyššie schopnosti agenta umožňujú abstrahovať od detailov nižších schopností agenta a tak zjednodušiť ich výber a postupnosť. Definovaním pohybov vyššej logiky je možné kombinovať pohyby vyššej úrovne, ktoré by sa mohli využiť pri plánovaní stratégie.

- **Chôdza** – predstavuje schopnosť dostať sa na určené miesto. Budú využité nižšie schopnosti agenta ako pohyb dopredu, či otočenie sa.
- **Vstávanie** – predstavuje schopnosť vstať z polohy ležmo. Budú použité nižšie schopnosti vstávania z chrbta a vstávania z brucha.
- **Kop do lopty** - predstavuje schopnosť kopnúť do lopty na stanovený cieľ. Týmto cieľom môže byť bránka, spoluagent, prípadne akýkoľvek bod na ihrisku. Použijeme nižšie schopnosti rôznych typov kopania do lopty a otáčania.
- **Vedenie lopty** - prestavuje schopnosť vedenia lopty jedným agentom. Bude pozostávať zo schopností kopu do lopty a chôdze.
- **Zorientovanie sa** – predstavuje schopnosť zistiť svoju vlastnú polohu na základe polohy bránok a polohu lopty. Táto schopnosť bude pozostávať zo schopnosti otáčania sa.
- **Blokovanie strely** – predstavuje schopnosť zablokovať strelu správnym postavením agenta a následným zablokovaním strely nižšími schopnosťami ako pádom, či rozkročením. Túto schopnosť bude využívať predovšetkým brankár.

3.1.6.1.3 Editor správania

V editore pohybov implementovali:

- Pridanie panela, ktorý slúži používateľovi na výber, či sa má kĺb ohýbať samostatne alebo spolu s jeho „bratom“ (symetrickým párovým kĺbom) a možnosť voľby, či sa má pohyb vykonať symetricky alebo po zrkadlovej trajektórii.
- Plug-in „Editor správania“, implementovaný v jazyku Java.

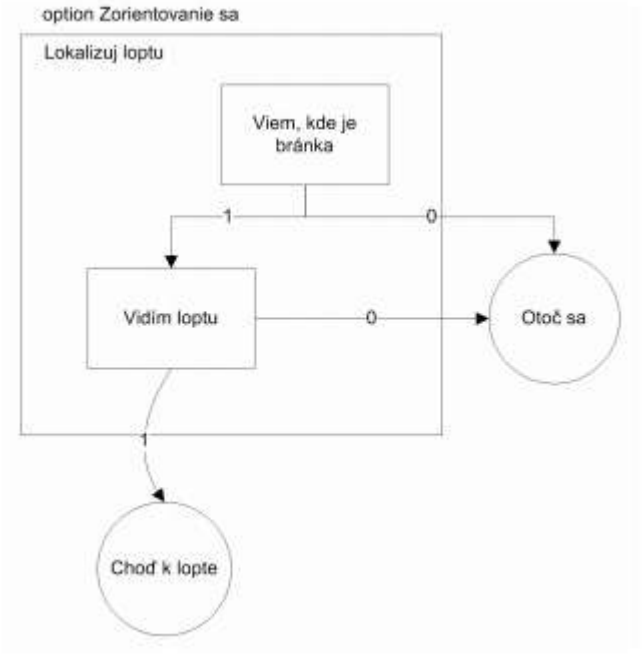
3.1.6.1.4 Definovanie vyššej logiky pomocou XABSL

Ako alternatíva produkčného systému plánovali použiť jazyk na definovanie vyššej logiky XABSL (Extensible Agent Behavior Specification Language), ktorý je vhodný na opísanie vyššej logiky hráča. K dispozícii je aj Java XABSL library. Opis správanie pomocou XABSL môže byť výhodné najmä, ak komplexnosť správania začne rásť a implementovanie v jazyku Java by začalo byť neefektívne.

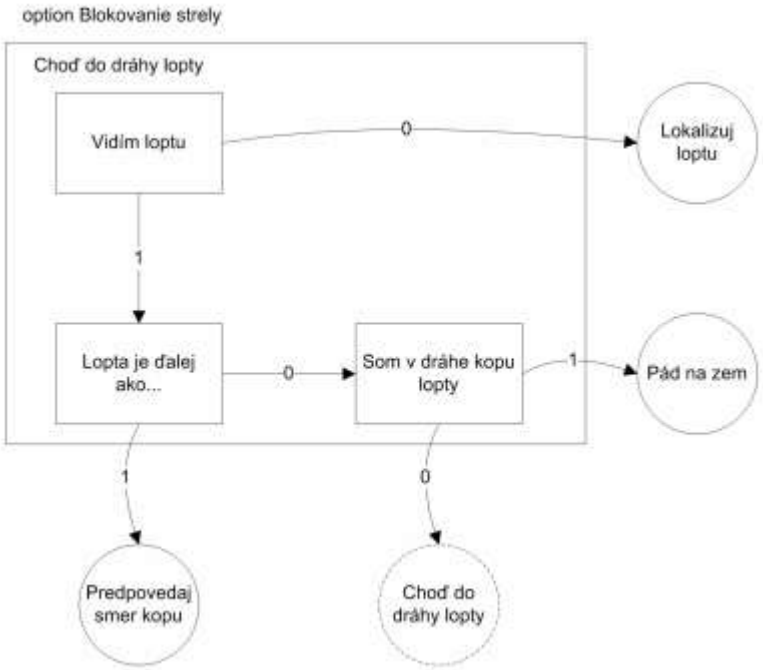
Agent v XABSL pozostáva z menších celkov – tzv. options, čo sú vlastne moduly správania, z ktorých každý je konečným automatom. V našom prípade pôjde o:

- **zorientovanie sa**
- **blokovanie strely**

Tieto sú zoskupené v strome, ktorého koncové uzly tvoria prvky nižšieho správania, ktoré vyvolajú požadovanú akciu. Prechody stavov realizujeme ako rozhodovacie stromy.



Obrázok 3.1: Pohyb vyššia logiky



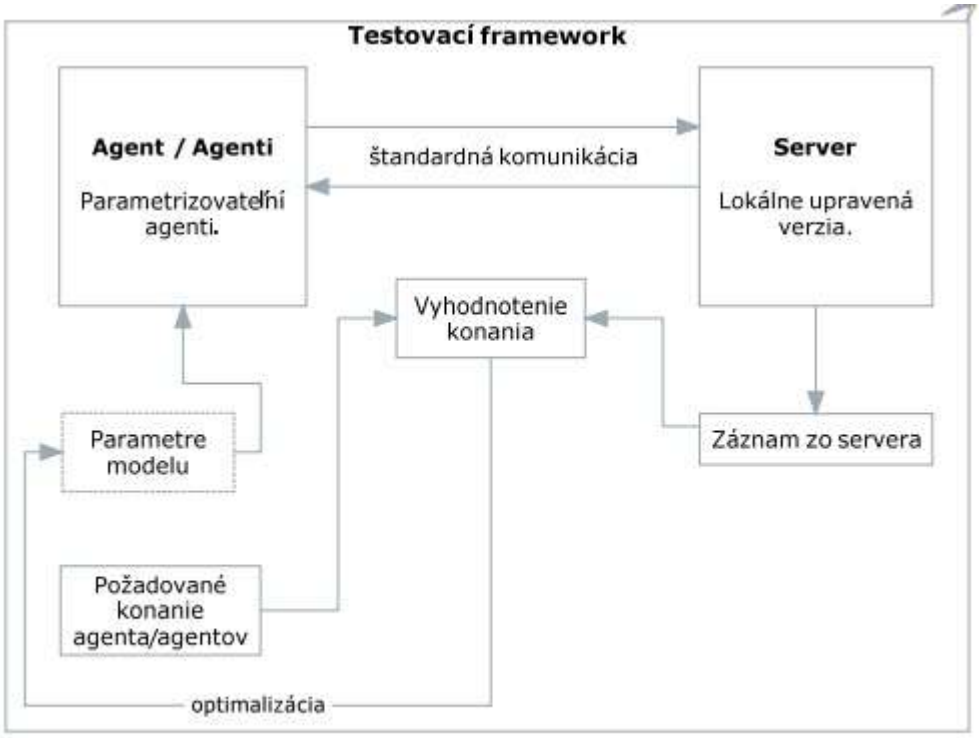
Obrázok 3.2: Pohyb vyššia logiky

3.1.6.1.5 Testovací framework

Z dôvodu zrýchlenia simulácie zasiahli do testovacieho frameworku a tak docielili efektívnejšie testovanie agenta.

Architektúra testovacieho frameworku

Nakoľko požadovaná funkcionalita pracuje na úrovni nad samotným agentom, testovací framework musí umožňovať vloženie agenta do systému. To platí v prípade testovania schopností nižšej úrovne. V prípade testovania vyššej logiky, musí framework umožňovať vloženie viacerých agentov do systému. Testovanie agenta vyžaduje špecifikovanie očakávaného výsledku. Následne po uskutočnení simulácie sa pomocou záznamu o zápase vyhodnotí, či výsledok dosiahol požadovanú úroveň. Ak nie, framework umožní technikami umelej inteligencie hľadať lepšie nastavenie parametrov. Vzhľadom na povahu problému bude vhodné použiť genetické algoritmy. Na obrázku je architektúra systému znázornená diagramom.



Obrázok 3.3: Architektektúra testovacieho frameworku

V nasledujúcej časti sú opísané netriviálne bloky, ktoré sa nachádzajú na obrázku:

- **Parametre modelu**. Blok špecifikuje jednoducho meniteľné parametre agenta, resp. agentov, ktoré budú môcť byť ovplyvňované v rámci optimalizácie. Parametre modelu predstavujú v skutočnosti komunikačný protokol pre použité genetické algoritmy a agenta.
- **Záznam zo servera**. Záznam zo servera slúži na získanie informácií o simulácii. Informácie, ktoré sa týmto spôsobom získajú, je následne potrebné pretransformovať do opisu, ktorý bude kompatibilný s opisom v bloku Požadované konanie agenta/agentov.
- **Požadované konanie agenta/agentov**. Požadované konanie je blok, do ktorého zadáva informácie používateľ. Tieto informácie sú opisom očakávaných udalostí, ktoré sa v bloku Vyhodnotenie konania budú porovnávať s informáciami získanými zoZáznamu zo servera.
- **Vyhodnotenie konania**. V predchádzajúcich opisoch bol uvedený účel blokuVyhodnotenie konania. Je potrebné uvedomiť si, že jeho zložitosťzávisí od miery rozdielu medzi informáciami získanými od používateľa a informáciami, ktoré získava blok zo Záznamu zo servera. Nakoľko informácie zo servera nie sú rýdzo deterministické,je potrebné vytvoriť opis, ktorý bude vždy môcť uvažovať s vopred stanovenou mierou nepresnosti.

3.1.6.1.6 Prototyp

Do prototypu boli zahrnuté nasledujúce funkcie:

- Implementácia testovacieho frameworku
- Pohyby agenta
- Úpravy v editore pohybov

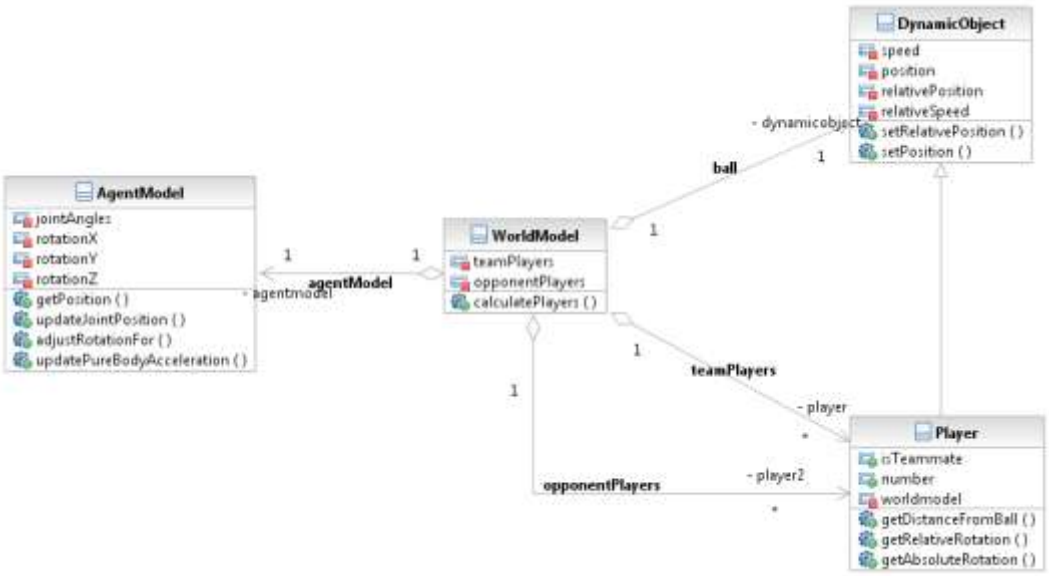
3.1.6.2 Analýza tímu High 5

3.1.6.2.1 Poloha ostatných hráčov

Tím High5 upravil parsovanie “see správ”, ktorých zdrojom je see receptor za účelom parsovania informácií o polohách ostatných hráčov na ihrisku.

Dáta o hráčoch sa ukladali do objektu PlayerData triedy Player. Trieda Player bola rozšírená aby uchovávala aj informácie o jednotlivých častí hráčov. Takto riešili získanie a uchovávanie informácií o polohe hráčov na ihrisku a o ich natočení.

Počítali aj vzdialenosť hráčov od lopty. Z absolútnej polohy hráčov (hlava) a lopty. Ďalej natočenie hráča voči lopte.



Obrázok49: Časť diagramu tried WorldModelu s dôrazom na hráčov

Obrázok 3.4: Časť diagramu tried WorldModelu s dôrazom na hráčov

Ako je vidno na obrázku tak upravená trieda Player už má implementované metódy:

- getDistanceFromBall() – ktorá používa metódu na určenie absolútnej polohy hráča na ihrisku
- getRelativeRotation()
- getAbsoluteRotation()

3.1.6.2.2 Testovací framework

Tím High5 vykonal dôkladný refaktoring testovacieho frameworku.

Rozdelili moduly na:

- Moduly vykonávajúce činnosti
- Implementačné moduly

Vyriešili spúšťanie pomocou jednej inicializačnej metódy, upravili logovanie aby bolo viac konzistentné a pribudol aj konfiguračný súbor.

Vďaka vyriešení obojsmernej komunikácie zo serverom je server informovaný o plánovaní hráčov (rozšírenie triedy AgentMonitor).

Tím vytvoril aj automatické spúšťanie hráča a servera, čo napomáha aj testovaniu. Vytvorené boli testy pohybov, ktoré dokážu ohodnotiť ich úspešnosť. Ďalej automatické vytváranie anotácií vytvorené v testovacom frameworku.

3.1.6.2.3 Pohyby

Optimalizácia kopu priniesla viacero úrovní sily kopu špičkou. Po analýze kopnutia špičkou po tíme Androids nebolo treba veľa zmeniť. Jediný problém bol s otáčaním hráča okolo svojej osi pri opakovanom vykonávaní pohybu. Problém bol odstránený. Tím vytvoril aj ďalšie 2 pohyby pre každú nohu, ktoré vychádzajú z kopnutia špičkou, líšia sa ale vo veľkosti napriahnutia a sile kopnutia.

Kopnutie bokom vylepšili väčším zohnutím kolena a členku pri kopaní, takto sa hráč nakloní viac dopredu a už s oveľa väčšou úspešnosťou triafa loptu.

Prerobili aj pohyb sadnutia, t.j. blokovania kopu sadnutím. Stabilitu pri tomto pohybe zlepšili nakláňaním trupu a úpravami záverečných fáz pohybu, ako napríklad zmenšením uhlu medzi nohami robota pri dosadnutí na zem. Vďaka úpravám sa zrýchlilo aj celkové vykonanie pohybu sadnutia. Tím vytvoril aj experimentálnu veľmi rýchlu verziu tohto pohybu. Tá ale nebola v konečnom dôsledku taká stabilná ako normálna verzia.

Optimalizovali a stabilizovali pohyb “walk_fine_fast2_optimized”. Pohyb zlepšili hlavne tým, že prerobili do “ľudskejšej formy”. Hráč viac používa kĺby a do istej miery imituje chôdzu človeka.

Vytvorili nový pohyb “walkback_slow”. Tento nový pohyb je vytvorený z pohybu “walkback3”, agent vykonáva ale menšie kroky smerom dozadu. Využíva sa najmä pri pohybu hráča okolo lopty.

Postavenie hráča za loptu. Vysokoúrovňový pohyb pohyb. Neúspešný návrh a implementácia pomocou vysokoúrovňových pohybov pohybov “WalkOld” (kráčanie za loptou) a “Lokalize” (lokalizovanie lopty).

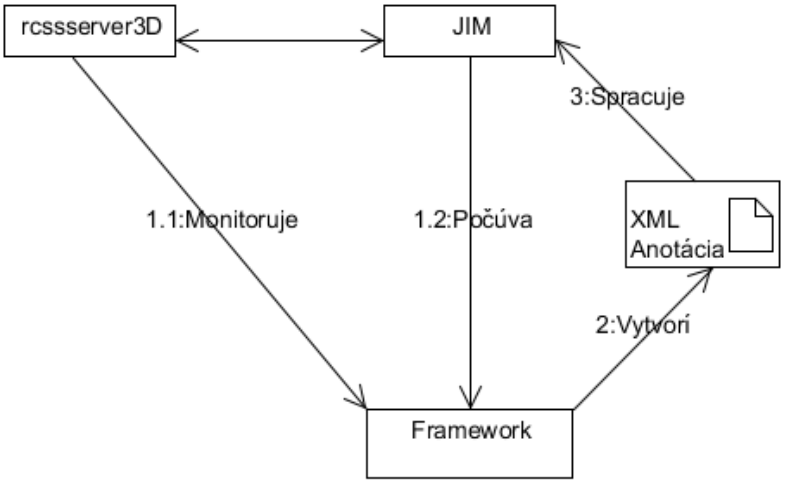
Vytvorili a optimalizovali 12 nízkoúrovňových pohybov. Ďalej vytvorili jeden komplexný vyšší pohyb, ktorého trieda je implementovaná tak, že ak z nej dedí iná trieda, musí implementovať jedine metódu pickHighSkill().

3.1.6.2.4 Anotácia pre pohyby

Tím navrhol a implementoval framework na tvorbu anotácií pohybov hráča. Anotácie slúžia na správne rozhodovanie pri plánovaní pohybov. Potrebovali k tomu počítať možnú pozíciu hráča z pohľadu hráča a nie z pohľadu ihriska.

Vypočítané hodnoty sa rozhodli uchovávať vo formáte XML.

Na obrázku je vidno proces spracovania údajov a vytváranie anotácií.



Obrázok 3.5: Proces spracovania údajov a vytváranie anotácií

Anotácie sa v druhom kroku rozšírili za účelom:

- lepšie reprezentovať vstupné a výstupné podmienky (poloha lopty)
- určiť bod najvyššej efektivity kopu do lopty

Vytvorili teda viac anotácií pre jeden pohyb a pridali rozptyl v akom sa lopta po vykonaní pohybu nachádza.

Zmenili reprezentáciu polohy lopty pomocou maximálnej, minimálnej a priemernej polohy, na kruhovú reprezentáciu.

3.1.6.2.5 Prerábanie GUI testovacieho frameworku

Navrhli a implementovali nové GUI s novými funkciami a vzhľadom. Doplnili možnosť pridávania nových hráčov, sledovania logov a možnosť ich konkrétneho výberu zo 7 kategórií (7 typov logovania).

Implementovali anotovací tab do GUI.

Vytvorili tretí tab v GUI na prenos dát sveta z Jim do testovacieho frameworku. Daný tab zahŕňa 2

podtaby, v ktorých je možné sledovať loptu alebo iného hráča z pohľadu agenta.

3.1.6.2.6 Ďalšie práce

Pomocou vykonanej analýzy hodnôt z akcelerometra a gyroskopu rozoznali hodnoty, ktoré je nutné sledovať pri navrhovaní pohybov robota, aby sa minimalizovali pády. Do budúcna sa to dá využiť ako pomôcka pri tvorení pohybov, ale aj ako navigácia pri adaptabilnej chôdzi.

Práca na vytvorení viacerých plánovačov.

3.1.6.3 Analýza tímu TÍM 17 ŽIJE...

3.1.6.3.1 Zistenie pozície stavu hráčov, ktorých agent vidí

Videný hráč stojí, ak je rozdiel Z súradnice hlavy a Z súradnice nôh väčší ako konštanta STANDING_LIMIT. Inak hráč leží.

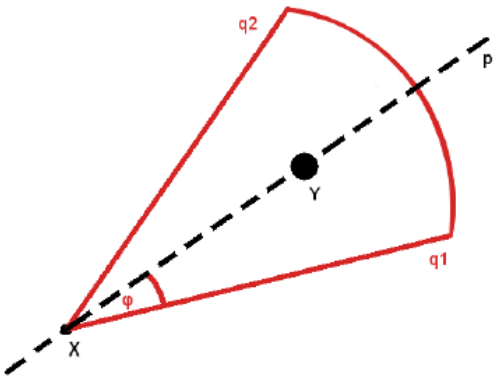
3.1.6.3.2 Vyhodnocovanie herných situácií

Pri brániacich a útočiacich situáciách boli pridané aj počty útočiacich a brániacich hráčov. Z polohy hráčov sa určuje, či je brániace mužstvo pod tlakom súpera alebo nie. Vlastník lopty je hráč s loptou od neho vzdialenou 1 meter. Hráč je pod tlakom ak je súper od neho vzdialený menej ako 2 metre.

3.1.6.3.3 Vytvorenie hernej formácie

Implementácia high skillu pre prejdenie hráč na určitú pozíciu vo formácii.

3.1.6.3.4 Zistenie vhodnosti prihrávky



X – hráč, Y – miesto kde má prihrávka skončiť.

|X,q1| = 1,5 x |X,Y|

Ak sa v červenom výseku nenachádza súper, tak je prihrávka vhodná.

3.1.6.3.5 Zadefinovanie high skillov

GetUp – hráč vstane ak leží na bruchu alebo na chrbte, ak leží inak tak vstáva ako z chrbta. Pri vstávaní z chrbta sa najprv preklápa na brucho.

Localize – keď hráč nevidí loptu dlhšie ako 5 sekúnd, tak sa otáča okolo svojej osi kým ju nenájde.

Walk – najskôr sa hráč k lopte natáča a kráča k nej, potom zaujíma vhodnú pozíciu.

Turn – smerovanie sa na cieľ a zároveň nevzdialenie sa od lopty.

Kick – pri vhodnej pozícii lopty a správnom natočení na bránku. Podľa vzdialenosti od lopty sa rozhoduje, či k nej ešte pristúpiť. Podľa polohy lopty sa rozhoduje, ktorou nohou vystrelí.

3.1.6.3.6 Prispôsobenie pohybov na zreťazenie

High skill sa môže nachádzať v jednom zo 4 stavov: initial, executing, finalizing, ended.

3.1.6.3.7 Plánovanie trajektórie

Trieda TrajectoryPlanner a zoznamy anotácií „rotate“ a „walk“ zabezpečujú výber pohybov vedúcich k želanému otočeniu a posunutiu sa po trajektórii.

3.1.6.3.8 Vylepšenie plánovača

Pri výpočte trajektórií existujú akceptovateľné odchýlky. Trieda Obstacles kontroluje prienik trajektórie s prekážkami, vypočítava bod obchádzania prekážky, kontroluje či sa bod nachádza vnútri hracieho poľa. Trieda *Trajectory* uchováva a sprístupňuje zoznam pohybov tvoriacich trajektóriu. Do triedy *Obstacles* bola pridaná metóda *getRealObstacles(),* ktorá z inštancie triedy *WorldModel* získava pozície všetkých hráčov na ihrisku a vracia ich ako zoznam prekážok. Trieda *TrajectoryRealTime* získava informácie pre výpočet trajektórie priamo z modelu agenta a modelu sveta.

3.1.6.3.9 Predikcia lopty

Predikcia pozície lopty sa počíta približne ako predikcia pozície hráča. Vyrátava sa vektor rýchlosti, pomocou ktorého sa počíta poloha hráča/lopty za X sekúnd.

3.1.6.3.10 Vytvorenie nových pohybov

Vytvorilo sa otočenie o 45 stupňov, pohyb otáčania sa okolo lopty, chôdza spojená s otáčaním, kop na diaľku.

3.1.6.3.11 Grafické zobrazenie v test frameworku

Do test frameworku bolo pridané grafické zobrazenie hry. V súčasnosti sa zobrazujú dáta z monitora pre všetkých agentov a loptu. Ďalej sa môžu zobraziť dáta z modelu sveta jedného alebo všetkých agentov, čo zahŕňa polohu a rotáciu agenta a polohu lopty.

3.1.6.3.12 Kop do lopty

Bol vytvorený silný priamy kop do lopty. Pozostáva z naváženia sa na jednu nohu, prikročenia k lopte druhou nohou, kopom prvou nohou a stabilizáciou hráča.

3.1.6.3.13 High skill – hranie futbalu

Implementovali sa metódy findTeammates pre nájdenie ostatných spoluhráčov, nearestToBall pre určenie či je hráč najbližšie k lopte čo vedie k priblíženiu sa k lopte, secondToBall ak je hráč druhý najbližší k lopte.

3.1.6.3.14 Zrýchlenie a spomalenie chôdze

Chôdza bola upravená tak, aby začala miernym prikrčením a zrýchľujúcou sa chôdzou, a končila spomaľujúcou sa chôdzou.

3.1.6.3.15 Vedenie lopty

Pri tomto pohybe sa hráč snaží zdvíhať nohy čo najmenej nad zem aby lopta neodskakovala ďaleko od agenta. Ďalej bol vytvorený high skill, ktorý zabezpečuje udržanie lopty stále pred agentom.

3.1.7 Analyzovanie RoboCup wikipedie (pp 1.7)

3.1.7.1 Analýza

Minuloročný tím vytvoril wikipediu k projektu RoboCup, ktorá popisuje najdôležitejšie časti a zhromažďuje najdôležitejšie údaje na jednom mieste. Wikipedia teda dokáže výrazne ušetriť čas pri hľadaní potrebných údajov.

3.1.7.2 Návrh

Každý člen tímu mal za úlohu analyzovať wikipediu RoboCupu a zvoliť si zopár zaujímavých skutočností, ktoré by bolo potrebné vedieť. Na nasledujúcom spoločnom tímovom stretnutí sa vytvorila diskusia, kde sa jednotlivé prístupy a informácie z wikipedie preberali a navzájom vymieňali medzi jednotlivými členmi tímu.

3.1.8 Analyzovanie zahraničných tímov (pp 1.8)

3.1.8.1 Analýza tímu Karachi Koalas

Karachi tím vznikol v polovici roka 2010 vďaka vedeckej spolupráci medzi technologickou univerzitou v Sydney (UTS) a inštitútu biznis administrácie, Karachi(IBA). UTS sa pravidelne zúčastňuje súťaže RoboCup už od roku 2003. V roku 2004 vyhrali súťaž RoboCup v Austrálii. Karachi Koalas skončil v prvej desiatke v súťaži RoboCup za rok 2012.

3.1.8.1.1 Vývojové prostredie

- C#/Mono
- TinMan pre komunikáciu zo serverom
- RoboViz na dynamické umiestňovanie lopty a hráčov, pohľad agenta

3.1.8.1.2 Architektúra

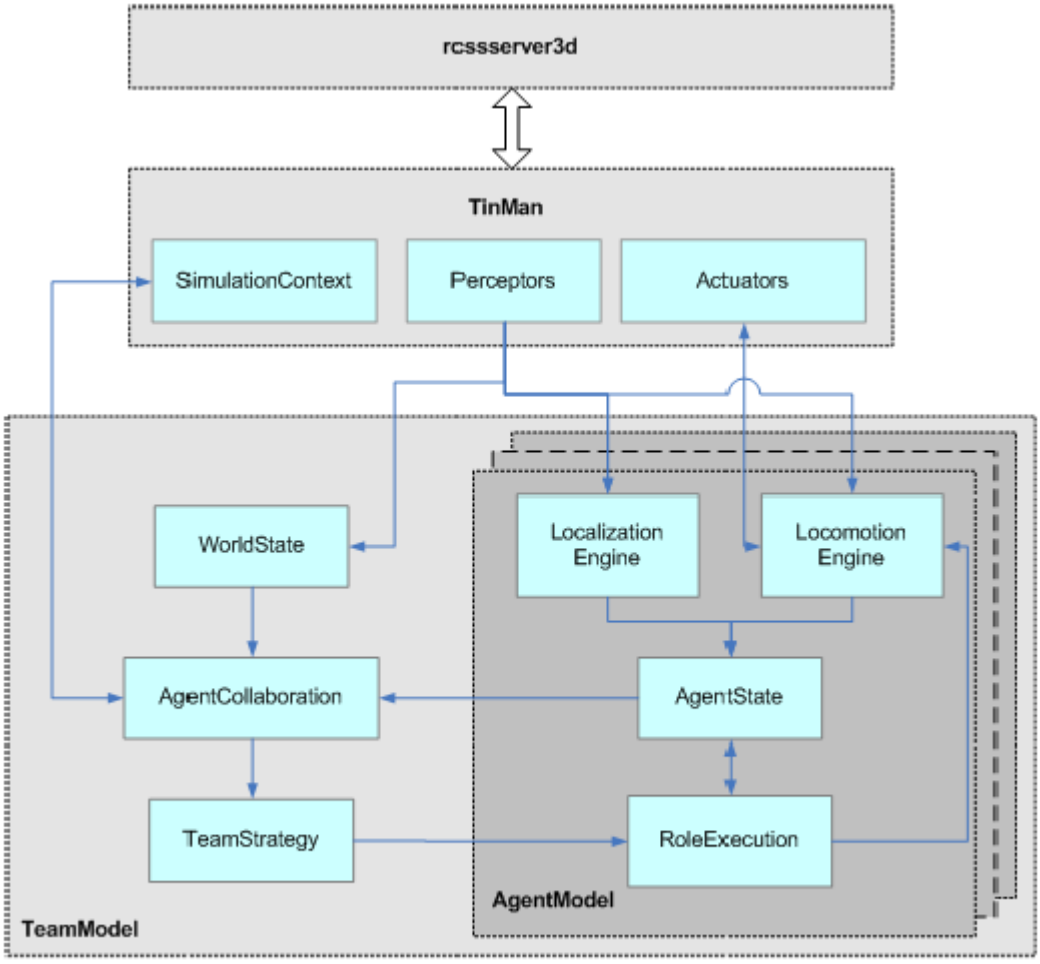


Fig. 1. Software Architecture

Obrázok 3.6: Softvérová architektúra

Ide o modulárnu architektúru. *TinMan* slúži ako rozhranie na RoboCup server *rcsserver3d*. Poskytuje abstrakciu aktivátorov(*Actuators*) a zachytávanie vnemov(*Perceptors*). *AgentModel* je zodpovedný za ovládanie jednotlivých agentov, ich stavov *AgentState*. Súčasťou agentovho modelu je *LocalizationEngine* a *LocomotionEngine.* Celkovú koordináciu medzi agentmi riadi *AgentCollaboration* a využíva pritom *AgentState* a *WorldState.* Aplikovaním heuristík rozhodne o stratégii. *TeamStrategy* má na starosti vykonanie určitej stratégie, spravuje formácie hráčov a preto poveruje *RoleExecution* zmenou roly jednotlivých hráčov. RoboCup server poskytuje priamu komunikáciu medzi agentmi cez rozhranie správ. *AgentCollaboration* používa *SimulationContext* na prijímanie a zasielanie broadcast správ.

3.1.8.1.3 Pohyblivosť

- Chôdza dopredu, dozadu a okolo
- Postavenie sa z chrbta a brucha, hádzanie brankára
- Kopy

3.1.8.1.4 Chôdza

Pre chôdzu bol navrhnutý spôsob inteligentného učenia sa pohybov kĺbov. Pohyby kĺbov boli modelované furiérovými radmi. Pri učení sa chôdze Nao robota a na optimalizáciu chôdze v simulačnom prostredí boli použité aj evolučné algoritmy. Pomocou NaoQi boli pozorovane ohyby kĺbov počas rôznych typov chôdzí. Zozbierané dáta použili v evolučných algoritmoch aby našli parametre pre furiérové rady. Rovnice, ktoré dostali boli rôznych zložitostí. Najzložitejšie mali 96 parametrov. Neskôr sa im počet týchto parametrov podarilo znížiť vďaka odhaleniu súvislostí pohybov jednotlivých kĺbov.

Fitnes funkcie evolučných algoritmov zahŕňali skutočnosti:

- Prejdená vzdialenosť robota
- Priamočiarosť jeho chôdze
- Stabilita robota pozorovaná gyroskopom

Pri otáčavých pohyboch bol dôraz na hľadanie najväčšieho počtu značiek v zadanom čase.

3.1.8.1.5 Postavenie sa z brucha, chrbta

Pomocou akcelerometra dokážu rozlíšiť polohu robota. So súradnicami x,y,z vedia rozlíšiť či robot padol na zem, či leží na chrbte alebo bruchu. Vždy keď robot padne na zem zavolá funkcie postavenia sa. Postavenie vyvoláva pohyby kĺbov v závislosti od stávania z bruchu, z chrbta.

3.1.8.1.6 Kopy

Tri typy kopov:

- Bočný kop – používaný ako nahrávka spoluhráčovi
- Uhlový kop – pre dribling alebo nahrávka hráčovi pred ním, v danom uhle
- Kop dopredu – pre strieľanie gólov alebo pasy

Kope sa pravou ako i ľavou nohou. Výber závisí na situácii.

3.1.8.1.7 Lokalizácia

Na lokalizáciu hráča používajú značky. Hráč má vnemové senzory a na základe týchto značiek sa určí poloha hráča. Vždy keď hráč vidí aspoň jednu značku tak vypočíta svoju polohu a orientáciu na ihrisku. Ak je v zábere iba jedna značka potom predpovedá pozíciu použitím rovníc a neskôr obnoví pozíciu hráča použitím priemeru súradníc. Ak nie je v dosahu žiadna značka, pohybovými rovnicami sú zistené približné súradnice. Počas chodenia hráč otáča hlavou zo strany na stranu aby vždy videl aspoň jednu značku. Ak hráč vidí loptu a je si istý jej pozíciou môže zasielať informácie iným hráčom čo loptu nevidia. Vyvinuli aj vizualizačný modul, ktorý odhaduje pozíciu hráča v poli a jeho pohyb v grafickom okne na overenie správnosti implementácie.

3.1.8.1.8 Stratégia

Dôraz riešenia bol kladený na lokalizáciu a pohyblivosť. Ich stratégia podporuje formácie tímov, prepínanie rol, vyhýbaní sa nebezpečných situácií. Vytvorili 4 roly, ktoré hráči môžu zastávať: brankár, obranca, záložník, útočník. Obrancovia sú rozložení na pravých a ľavých. Rola záložníka je dynamická a môže sa po splnení určitých podmienok stať útočníkom. Ostatní hráči sú predvolene záložníci.

3.1.8.1.9 Zdroje

- http://karachikoalas.iba.edu.pk/files/karachikoalas_TDP.pdf

3.1.8.2 Analýza tímu rUNSWift

Tím rUNSWift vznikol v roku 1999. V rokoch 1999 až 2006 sa zúčastňoval svetovej súťaže v najvyššej kategórii 4-Legged League, kde dosahoval výborné výsledky. Medzi prvými troma priečkami sa umiestnil v každom roku. Prvé miesto tím dosiahol v rokoch 2000, 2001 a 2003. Od roku 2008 súťažili v kategórii Standard Platform League, v ktorej boli štvornohý roboti nahradený robotmi dvojnohými. V roku 2010 získali druhé miesto a v roku 2012 obsadili tretiu prečku. Tím rUNSWift patrí pod univerzitu The University of New South Wales, ktorá sídli v Austrálii.

3.1.8.2.1 Výskum

Dlhodobým cieľom je vývoj všeobecne použiteľného inteligentného systému, ktorý by učil vykonávať mnoho rozličných úloh samostatne len za pomoci interakcie v prostredí.

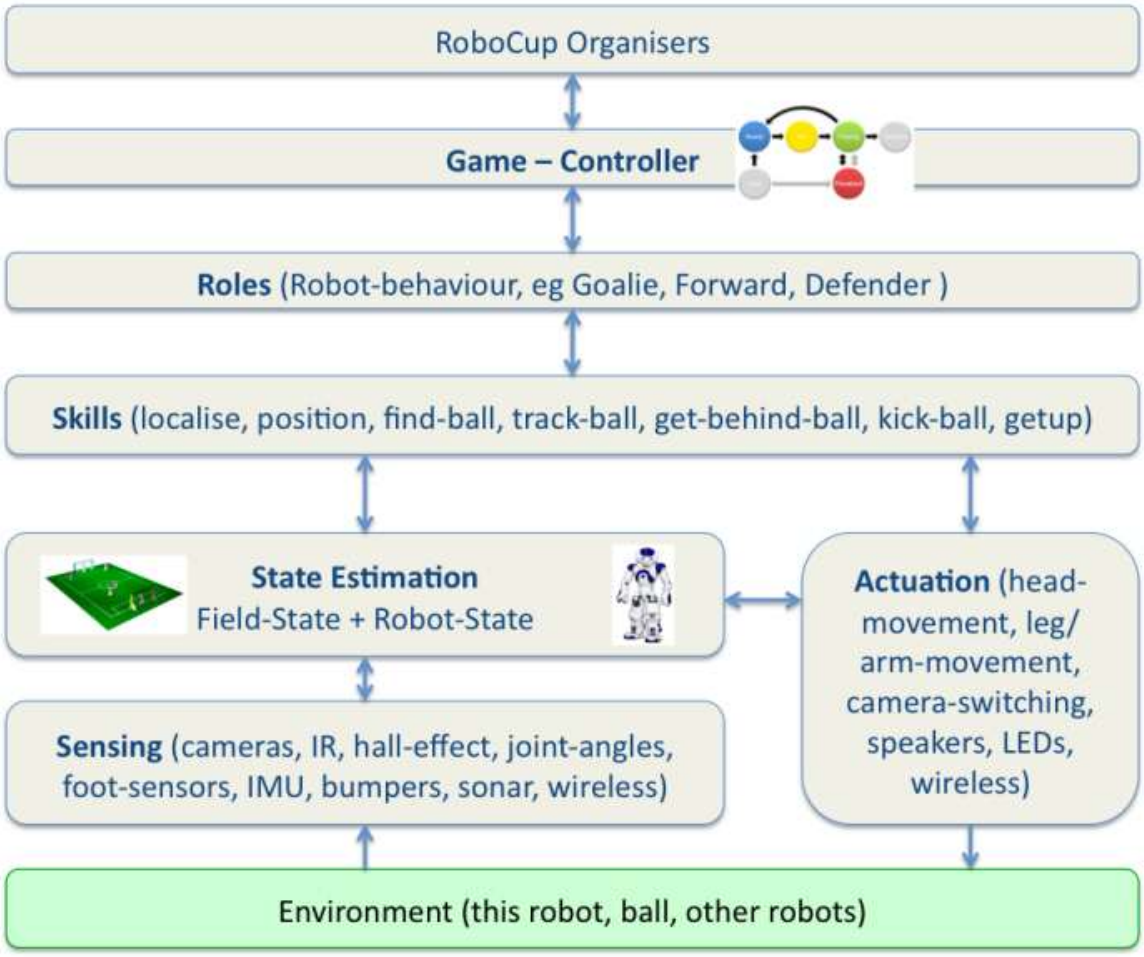
Hlavnými všeobecnými výskumnými cieľmi tímu rUNSWift v SPL (Standart Platform League) je:

- Ďalej rozvíjať rozumové metódy, ktoré zahŕňajú neurčitosť a real-timeové obmedzenia. A integrovať ich so štatistickými metódami používanými pri vnímaní.
- Vyvíjať metódy pre používanie odhadov neurčitosti a riadiť tak budúce rozhodovanie tak, aby eliminovala neistota a neurčitosť.
- Rozšíriť tieto metódy na multi-robotovú spoluprácu.
- Používať symbolickú reprezentáciu ako základ pre interakciu humanoidného robota.
- Vyvíjať učiace algoritmy pre hybridné systémy, ako je využitie znalostí logických obmedzení na vyvarovanie sa vyhľadávaniu pokus-omyl.
- Vyvíjať high-level symbolický jazyk pre robotov, ktorý poskytne abstrakciu pre veľké množstvo úvah, plánovanie a učiace techniky tza účelom zjednodušenia programovania robotov.

3.1.8.2.2 Robotická architektúra

Robotická architektúra je úlohová hierarchie pre multi-agentný tím štyroch Naos (Naos je najnovší druh robota, ktorý sa používa na svetových turnajoch). Tím používa centrickú architektúru s chybovou toleranciou. To znamená, že každý robot môže mať mierne odlišný pohľad na svet a preto aj inú rolu v tíme. Tento prístup má tú výhodu, že poskytuje čiastočnú redundanciu v prípade, že niektorý robot prestane pracovať alebo je diskvalifikovaný.

Herný ovládač vyvolá na koreňovej úrovni high-level stavy pre začatie hry. Na nižšej úrovni, generátor chôdze vykoná fázy chôdze, ktoré vyvolajú stav prechodu tvoriaci pohyb robota.



Obrázok 3.7: Robotická architektúra

3.1.8.2.3 Videnie

Systém videnia vyvinul tím ešte v roku 1999. Od začiatku používali jednoduchý výučbový systém na trénovanie systému na rozpoznávanie farieb. V roku 2001 začali používať strojové učenie na budovanie rozoznávača rozhodovacieho stromu. To sa ukázalo veľmi dôležité, pretože v súťažiach by ich predchádzajúci systém videnia nebol dostatočný.

V posledných rokoch bol ich systém aktualizovaný na rozpoznávanie ELD značenia a menej sa orientovali na farbu použitím hranových funkcií. V súčasnosti skúmajú *foveated* videnie a virtuálne kmitanie za účelom maximalizácie výpočetných zdrojov dostupných v Nao-vi. Pre eld čiary a eld hrany vyvinuli senzorový model, ktorý poskytuje viac hypotéz pre novú lokalizáciu.

3.1.8.2.4 Lokalizácia

V roku 2000 bolo prvý krát na súťaži použitá Kalman Iter lokalizačná metóda, ktorá bola v nasledujúcich rokoch vyvíjaná. Vďaka výhodnej lokalizácii a pohyblivosti v roku 2000 tím nedal nikdy menej ako 10 gólov a dostal maximálne jeden gól. Od roku 2000 bol systém lokalizácie prepracovaný a zahŕňa multi modálne distribuovanie dát cez sieťovaných robotov. V roku 2006

upustili od zdieľania informácii medzi robotmi ale zaoberali sa nimi ako s jedným tímom viacerých robotov. To umožňovalo spracúvať viac hypotéz a taktiež používať loptu pre získanie informácii ohľadne lokalizácie.

3.1.8.2.5 Pohyblivosť

V roku 2000 bol predstavený pohyb UNSW (skratka univerzity), ktorý sa používal na súťažiach. Kľúčovou myšlienkou tohto pohybu bolo popísať trajektóriu labky robota jednoduchými geometrickými prvkami, ktoré boli parametrizované. Vďaka tomu sa stali hráči rýchlejší ako mali konkurenčné tímy. Od vtedy mal takmer každý tím na turnaji podobný štýl pohybov, ktorý vychádzal z kódu rUNSWIFT. Zmena prišla v roku 2003, keď tím požil strojové učenie na vylepšenie robotovej chôdze a tým dosiahol moc rýchlejšie pohyby. Od vtedy veľa tímov používalo ich systém na strojové učenie na vylepšovanie chôdze svojich hráčov.

Náš súčasný výskum je zameraný na výučbu všestrannosti bipedálnych pohybových ovládačov použitím hierarchického posilňujúceho učenia.

3.1.8.2.6 Zdroje:

- http://cgi.cse.unsw.edu.au/~robocup/2011site/reports/Robocup2011rUNSWiftTeamDescription.pdf
- http://www.cse.unsw.edu.au/help/students/robocup/

3.1.8.3 Analýza tímu beeStambul

BeeStambul je projektom laboratória umelej inteligencie a robotiky na Technickej univerzite v Istambule. Na RoboCup projekte sa zúčastňujú od roku 2009, súťažia od roku 2010 a majú za sebou mnoho úspešných víťazstiev.

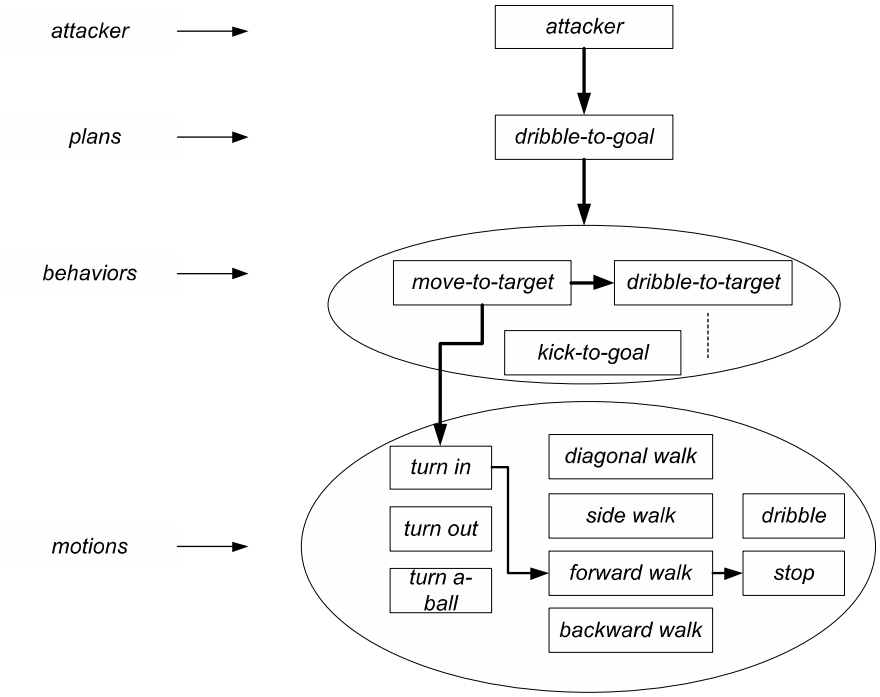
3.1.8.3.1 Návrhy

Pre pohyb v koronárnej rovine v prvotnej implementácií používali model Fourierovho radu. Táto implementácia zahŕňala sedem hlavných pohybov, špeciálne prechodové funkcie a hladký prechod medzi dvoma ľubovoľnými pohybmi. Tieto prechody pôsobili ale príliš veľké oneskorenie v reakciách agenta.

V novom návrhu implementovali statické a dynamické pohyby. Statické boli postavenie sa hráča alebo hľadanie lopty, dynamické pohyby sa generovali dynamicky podľa cieľa hráča. Dynamické plánovanie je založené na rôznych typoch chôdze, otáčok a zarovnaní. Parametre optimalizujú pomocou evolučnej stratégie CMA (Covariance Matrix Adaptation).

3.1.8.3.2 Správanie

V stratégií tímu je vrstvený návrh výberu pohybov, správania a plánovania (layered motion selection architecture). Táto hierarchia dosiahla vysokú modularitu a ľahkú údržbu. Tím vytvára pohyby a plánovanie na vyššej úrovni pomocou pohybov nižších úrovní. Príklad na schéme č.1.



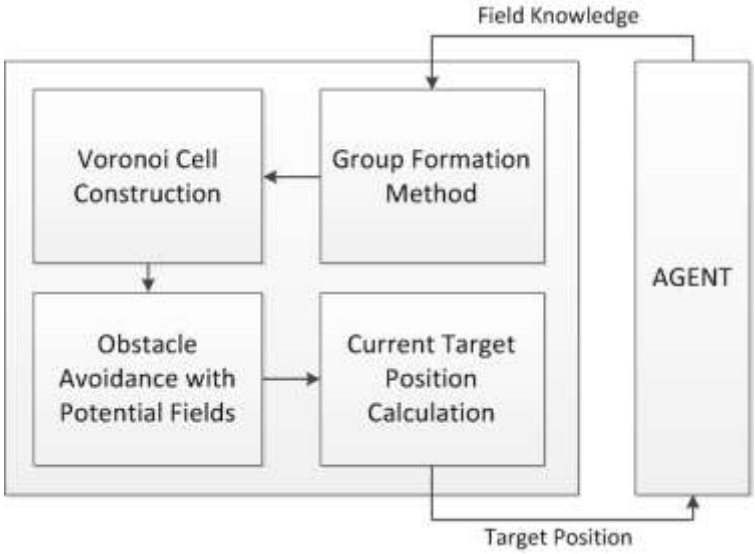
Obrázok 3.8: Pohyb dribble-to-goal

3.1.8.3.3 Tímová stratégia a plánovanie

Tímová stratégia zahŕňa štyri po sebe idúce procesy pre určenie cieľa hráča. Na začiatku sa formulujú dve skupiny, útočníci a obranári.

Majú tri rôzne plánovače pre štyri roly. Forward plánovač vedie loptu bez ohľadu na spoluhráčov a vyberá medzi akciami vedenia lopty ďalej alebo kopania na bránu. Plánovač brankára sa snaží tvoriť prekážku pre protihráčov. Finálny plánovač je používaný aj strednými aj útočiacimi hráčmi, rozdielne je iba kalkulovanie na základe cieľa. Plánovače sú aktivované na základe pridelenia roly pre hráča.

Hlavný modul pre tímovú stratégiu je na obrázku č.2.

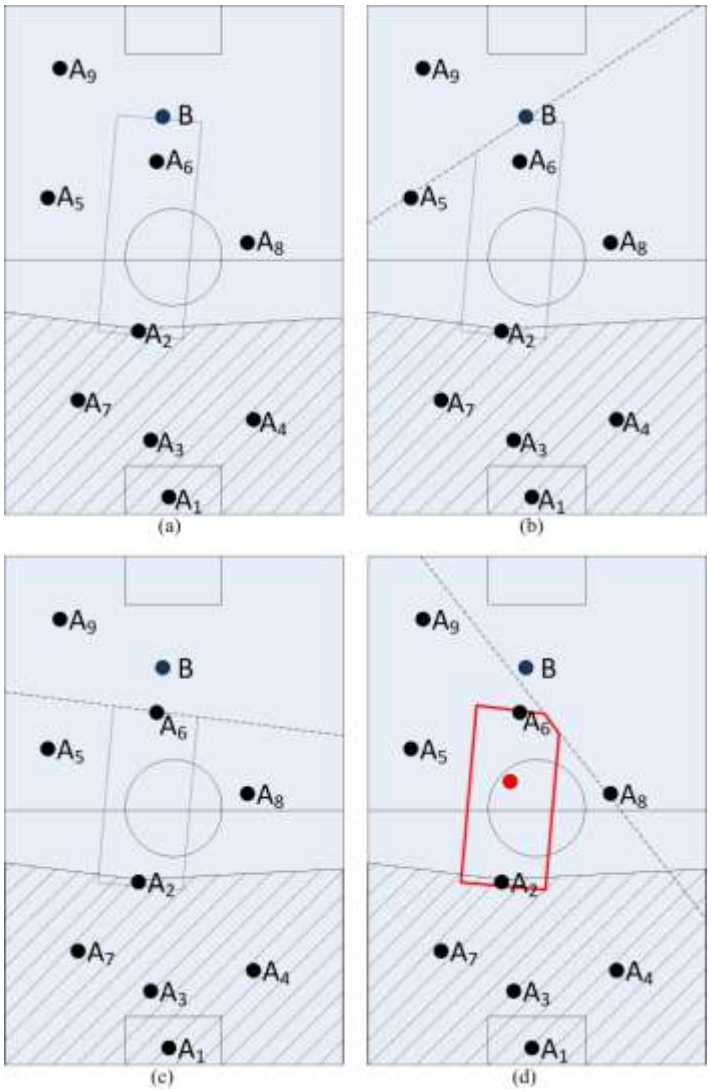


Obrázok č. 1

Počet a pomer brániacich a stredných hráčov je určený pomocou skupinového formulovania na základe stavu hry.

Obrancovia a stredné hráči sa polohujú tak, aby útočníkov mohli udržiavať čím viac vpredu. Na toto používajú výpočet svojich teritórií, ktoré si chránia. Tieto teritória hráčov sa môžu aj prekrývať.

Stredoví hráči si rátajú svoje teritórium na základe polohy lopty a spoluhráčov na ktorých vidia. Obrázok č.3. znázorňuje modifikovanie základného teritória stredného hráča na základe jeho spoluhráčov.



Obrázok č. 2

Obrázok č. 3 znázorňuje zúženie teritória hráča A2 na základe priesečníkov cez jeho spoluhráčov (A5, A6, A8) a jeho základného teritória.

Svoje teritórium si určia aj brániaci hráči podobne. Oni si ale vytvárajú svoje základné teritórium na základe stredového bodu na línie medzi loptou a stredom ihriska.

Po vykonaní týchto výpočtov sa hráči posunú na stred svojich teritórií.

3.1.8.3.4 Zdroje

- B. Demirdelen, B. Toku, O. Ulusoy, T. Sonmez, K. Ayvaz, E. Senyurek, and S. Sariel-Talay , beeStanbul RoboCup 3D Simulation League Team Description Paper 2012, url: http://air.cs.itu.edu.tr/publications-1/beeStanbul_TDP2012.pdf

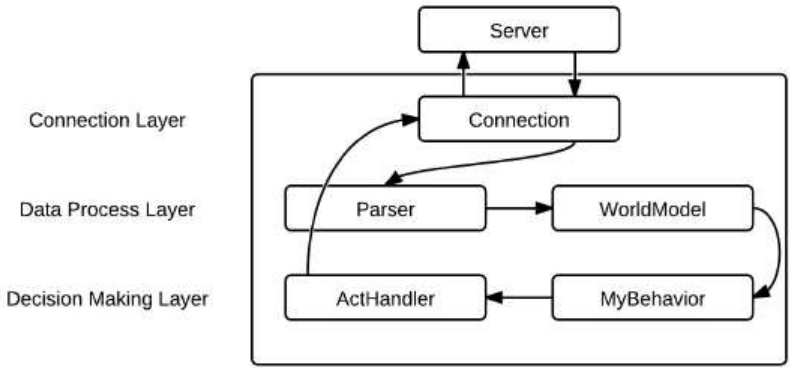
3.1.8.4 Analýza tímu Nexus3D

Tento tím patrí medzi menej úspešné v rámci medzinárodných súťaží RoboCup 3D. Pochádzajú z iránskej Ferdowsi University of Mashad. Začínali so simuláciou 2D futbalu a od roku 2004 sa snažia uspieť v poli 3D.

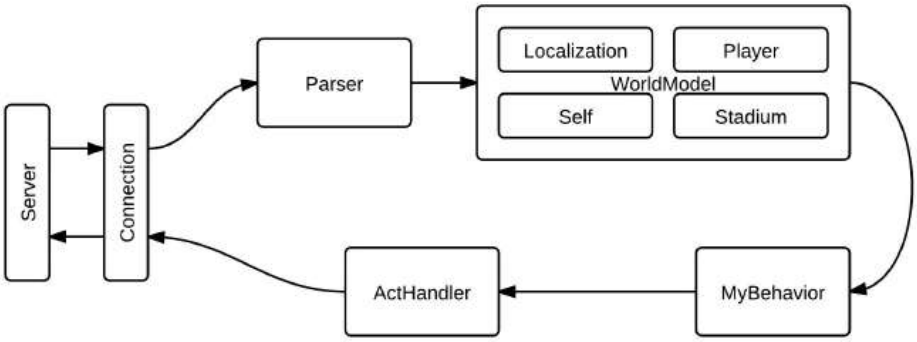
3.1.8.4.1 Architektúra

3 vrstvy:

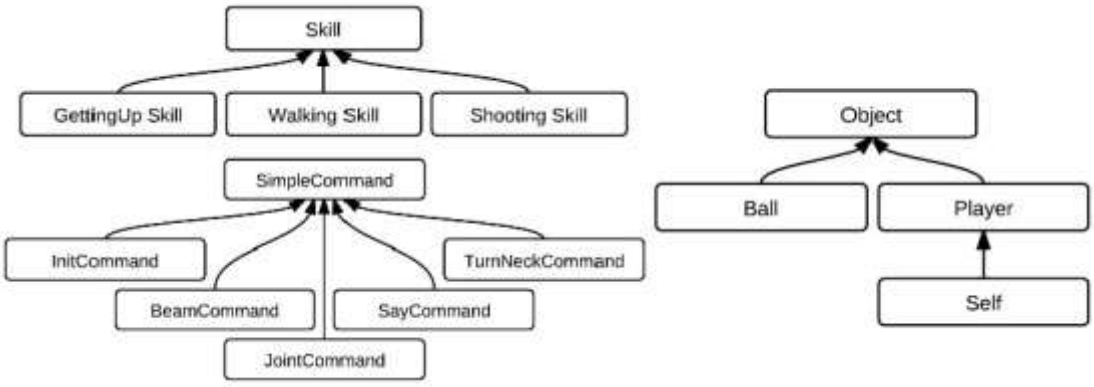
- Communication – cez perceptory a effectory
- Data - zparsované dáta sú posunuté do WorldModel (informácie o prostredí), ktorý sa delí na:
    - Localization - pozície
    - Stadium – všeobecné info o zápase
    - Aget
    - Other players
- Decision making – rozhodovanie podla informácií vo WorldModel, výsledok je Behavior pozostávajúci zo SoccerCommands



Obr.1: Vrstvy

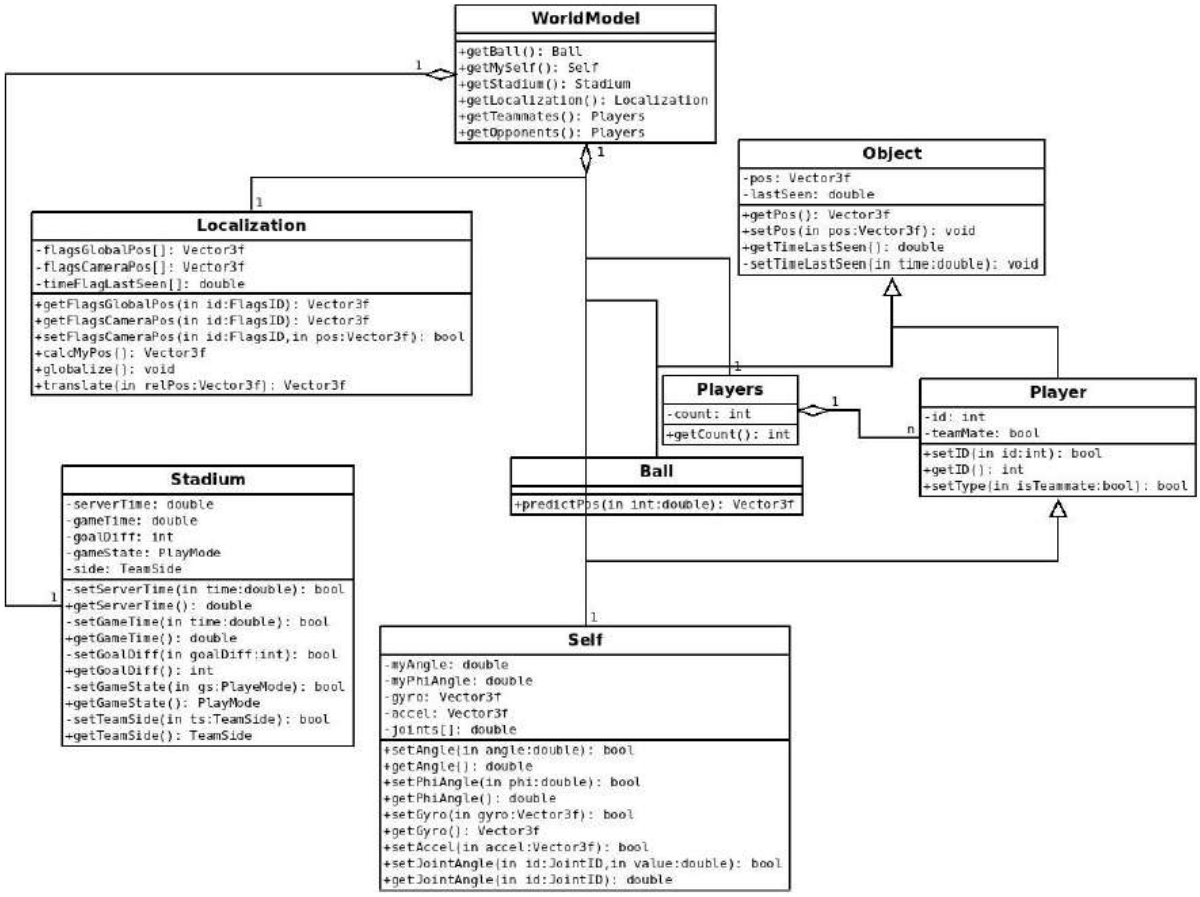


Obr.2: Data-flow diagram kodu



Obr.3: Dedenie niektorých tried

3.1.8.4.2 WorldModel



Obr.4: Štruktúra modelu sveta

3.1.8.4.3 Lokalizácia

- relatívna pozícia (relatívna k hráčovi) a globálna pozícia (relatívna k stredu ihriska)
- viditeľnosť dvoch vlajok postačuje na vytvorenie jednoduchej mapky ihriska s pozíciou hráča

3.1.8.4.4 Príkazy, schopnosti a správanie

- Command – SoccerCommand pozostáva zo SimpleCommand
- Skill – sada príkazov ústiacich do akcie (chôdza,...):
  - Prepare – vykonanie príkazov k príprave hráča na vykonanie podstaty skillu
  - Do – vykonanie príkazov podstaty skillu
  - Stop – vykonanie príkazov ktoré ukončia vykonávanie skillu (po skončení skillu alebo počas neho)
- ActHandler – vykonáva skill, ak má vykonať skill a iný sa už vykonáva, tak ten aktuálny stopne a až potom vykoná ďalší
- Behavior – sada skillov

3.1.8.4.5 Zdroje

- http://nexus.um.ac.ir/sources/NexusEN.pdf

3.1.8.5 Analýza tímu magmaOffenburg

Tento tím začal s 2D simuláciou ligy, kde ešte vystupoval pod menom magmaFreiburg. Najväčší úspech mal v roku 1999 kde na svetovom šampionáte obsadil 2. miesto. V roku 2009 prešiel k 3D simulácií. V tom roku sa aj premenoval na tím magmaOffenburg. Pod týmto menom vystupuje až dodnes. Tím magmaOffenburg je tímom Hochschule Offenburg University of Applied Sciences.

Na tvorbu správania robota majú vytvorený rámec, ktorý sa skladá z troch časti. Každý pohyb je rozdelený do pohybových fáz, ktoré pozostávajú z jednotlivých pohybov kĺbov. Medzi špeciálne funkcie patri automaticky prevod pohybu pravej časti tela do pohybu ľavej časti tela. Každý pohyb kĺbu ma definovanú rýchlosť, ktorou sa nemusí nutne pohybovať až do konca pohybovať fázy.

Keďže tím ma vytvorený dynamicky model robota, na jeho primerané pohyby použili inverzne kinematické výpočty. Na tento účel každá časť tela, dynamicky poskytuje metódu pre vytvorenie Jacobiho matice od reťazca časti tela, po koreň časti tela. Časť tela inštancie poskytuje niečo potrebné na vykonanie typického základného cyklu inverznej kinematickej metódy medzi žiadajúcou Jacobiho maticou a vykonávaným delta pohybom na príslušný kĺb. Takýmto spôsobom rôzne kinematické metódy, môžu byť použite na optimalizáciu pozície/orientácie konkrétnych časti tela, inštancia modelu tela reprezentuje ľubovoľný fyzicky model robota k cieľovej pozícii/orientácii v priestore. Aktuálne úsilie sa vynakladá na vyšetrenie rôznych kinematických metód a ich použiteľnosť v humanoidných robotoch. Skutočná sila inverznej kinematickej metódy spočíva z odobratia fyzikálnych vlastnosti z robota. Tak môžu cieľovú pozíciu a orientáciu vypočítať dynamicky pomocou ďalšieho senzora informácii a tak odovzdať tieto informácie inverznému kinetickému rámcu, ktorý adekvátne presunie robota.

3.1.8.5.1 Zdroje

- http://robocup.hs-offenburg.de/html/index.htm

3.1.8.6 Analýza tímu Austin Villa

Tím z Texaskej univerzity, ktorý sa sútredí na trening, hranie, všesmerovú chôdzu, rozpoznávanie a učenie sa farieb atď. Svojich hráčov rozdelujú na dve skupiny a to trénera a hráčov.

Tréner – pozoruje hru protivníka, učí sa a dokáže vyprodukovať strategiu.

Hráč – akceptuje trénerove taktiky a vykoná ich.

Majú aj dve stratégie. Prvá je taká, že jeden tím sa bráni a snaží si udržať loptu a druhý sa snaží ju zobrať a dať gól. Druhá stratégia je, že sa tímy navzájom snažia prekonať súperovu obranu a dať gól.

V Nao lige definujú 4 hlavné problémy videnie, lokalizácia, pohyb a koordinácia. Tieto problémy popisujú a snažia sa riešiť. Vychádzajú z obohacovaného učenia sa a predpokladajú, že robot by mal byť schopný sa viac naučiť zo skúseností z reálneho sveta ako z naprogramovania jeho vedomostí. Obohacované učenie ešte obohacujú o rozhodovacie stromy. Agent sa pomocou ich prístupu rýchlejšie učí ako pri bežných voľných učiacich modeloch.

Algoritmus RL-DT (reinforcement learning and decision tree):

RL-DT(RMax, s)
1: A Set of Actions
2: S Set of States
3: 8a 2 A : visits(s, a) 0
4: **loop**
5: a argmax$_{a' \in A}$Q(s, a′)
6: Execute a, obtain reward r, observe state s′
7: Increment visits(s, a)
8: (PM,RM, CH) UPDATE-MODEL(s, a, r, s′ , S,A)
9: exp CHECK-POLICY(PM,RM)
10: **if** CH **then**
11: COMPUTE-VALUES(RMax,PM,RM, SM,A, exp)
12: **end if**
13: s s′
14: **end loop**

Prišli s novým prístupom k lokalizácii hráča na ihrisku. Využívajú pri tom Microsoft Kinect RGB-D Sensor, čo je lacné prenosné a rýchle riešenie. Používajú to hlavne na zistenie polohy lopty a robota.

Systém je pre nich ľahko nastaviteľný a nevyžaduje nové senzory. Jednou z hlavných úloh je transformácia pointcodového výstupu z kinect senzora do mapy ihriska. Ďalším kľúčovým doplnkom je rozpoznanie farieb na ihrisku. Následná identifikácia, ktorým smerom sa má robot vybrať

vychádza zo získaných dát, ale aj z detekcie objektov na ihrisku. Robot musí vedieť identifikovať druhého robota a loptu. Napríklad pri detekcií lopty sa snažia zachytiť skupinu oranžových bodov v určitej vzdialenosti od seba. Pri testoch dokázal identifikovať druhého robota na skoro 96%.

Zaujímavosťou na záver je aj ich oblasť záujmu v štvornohej lige s robotom Sony Aibo. Štvornohí roboti v podobe psov sú obratnejší a majú odlišnú škálu pohybov. Vďaka ich konštrukcií sa nemusí riešiť koordinácia.

3.1.9 Analyzovanie robotov (pp 1.9)

3.1.9.1 Analýza robota ASIMO

ASIMO = Advanced Step in Innovative Mobility od Honda Motor Company. Je to prvý humanoidný robot, ktorý vie nezávisle chodiť a kráčať po schodoch. Dokáže rozumieť predprogramovaným gestám, hovoreným príkazom, rozoznávať hlasy a tváre. Je to robot pomocník.

Ľudia z Hondy študovali pohyby, hmyzu, ľudí, horolezca s protézami, aby lepšie pochopili fyziológii chôdze. Sledovali prenášanie váhy pri chôdzi, špeciálne pohybmi rúk. Všimli si úlohu palca na nohe pri zachovávaní rovnováhy.

Asimo má bočné, kolenné a chodidlové kĺby. Výskumníci nazvali kĺby stupňami slobody. Asimo má 34 stupňov slobody po celom tele.

Má rýchlostný senzor a gyroskopový senzor, ktoré:

- zaznamenávajú polohu a rýchlosť tela
- prenášajú prispôsobenie rovnováhy do centrálneho počítača

Tieto dva senzory pracujú podobne ako naše vnútorné ucho pre udržiavanie stability a orientácie. Asimo má tiež aj podlahové povrchové senzory v nohách a šesť ultrasonických senzorov v strede tela. Slúžia na detekovanie okolitých objektov a ich porovnávanie s dátami máp uloženými v pamäti.

Asimo má senzory zisťujúce natočenie kĺbov a osové silové senzory vnímajúce tlak vykonávaný na predmety.

Vďaka svojim otáčacím schopnostiam nemusí počas presúvania zastavovať a meniť smer alebo robí to počas toho. Na dosiahnutie takejto vyspelej chôdze skúmali vedci inerciálne sily počas chôdze. Napríklad sila gravitácie, sila rýchlosti chôdze, sila chodidla pôsobiaca na zem a pod. Pomocou ZMP „zero moment point“ sa tieto sily vyrovnávajú. Asimov postoj je ovládaný troma ovládaniami:

- floor reaction control – vyrovnáva sa s chybami povrchu
- target ZMP control – vyrovnáva časti tela, aby robot nepadol
- foot-planting location control – ovláda dĺžku krokov vzhľadom na pozíciu a rýchlosť tela

Asimo nielen predchádza pádu, ale vďaka jeho hladkému tempu sa vie otáčať bez zastavovania. Používa technológiu „predictive movement control“, ktorá zabezpečuje ako ďaleko do zatáčky sa

posunie ťažisko a ako dlho tam bude posunuté. Všetko toto sa deje v reálnom čase. Počas chôdze si nastavuje nasledovné:

- dĺžku krokov
- pozíciu tela
- rýchlosť
- smer krokov

Vďaka tomuto dokáže bežať rýchlosťou až 6 km/h. Počas behu sa dokáže ocitnúť vo vzduchu až 0,08 sekundy. Aby počas toho nestratil kontrolu, nestočil sa vo vzduchu alebo nepadol pri dopade, má Asimo v torze jeden stupeň slobody.

Asimo má v hlave dve kamery na mieste očí a používa stereoskopický pohľad.

3.1.9.1.1 Zdroje

- http://science.howstuffworks.com/asimo1.htm

3.1.9.2 Analýza pohybov

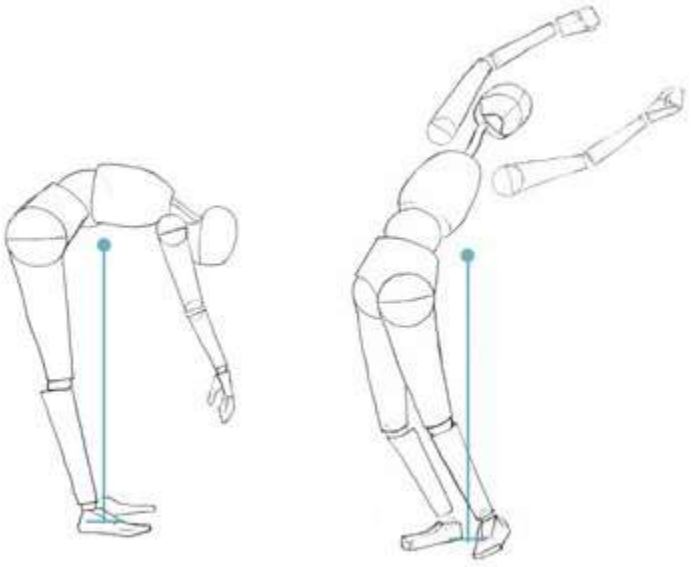
Najbežnejšie metódy pohybu, ktoré môžu byť rozdelené do 4 kategorii:

1. trajectory-based approaches (prístupy založené na trajektórii)

   Pozostáva z nájdenia kinematických trajektórií a použitia stabilizačných kritérií na zaistenie stabilnej chôdze. Stabilizačné kritéria :

   - Center of Mass – ťažisko systému častíc je bod v ktorom sa hmotnosť systému sprava ako keby bol koncentrovaný. Inak povedané ťažisko je definovane ako umiestnenie váženého priemeru jednotlivých hmotných bodov systému –

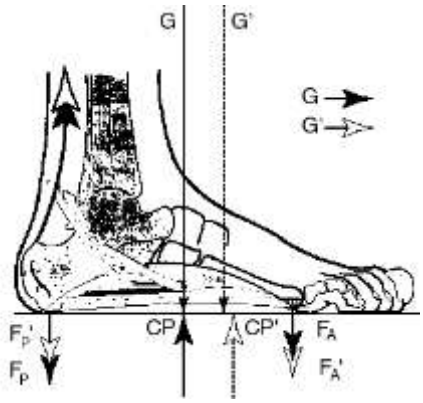
$$p_{CoM} = \frac{\sum_i m_i p_i}{M}$$



Kde, M = suma m_i, celková hmotnosť systému, m_i – množstvo itej častice, p_i - ťažisko

- Center of pressure – centrum tlaku – veľa humanoidov je vybavených senzorom krútiaceho momentu na jeho nohách. Centrum tlaku je výsledkom vyhodnotenia týchto senzorov a je definovaný ako bod na zemi kde pôsobí vyplývajúca reakcia pozemných síl

-



- Zero moment point – bod na zemi v ktorom mement všetkých aktívnych síl je nulový

2. virtual model control (virtuálny model riadenia)

Virtual Model Control (VMC) je rámec založený na heuristike, ktorý používa virtuálne komponenty, ako sú klapky pružiny, alebo hmotnosť ku generovanie spoločných momentov ktoré kontrolujú stabilitu a rýchlosť robotov. Vytvorene komponenty majú rovnaký efekt ako keby boli skutočné.

3. passive-dynamic walking (pasívna-dynamická chôdza)
    - je založená na obrátenom kyvadle (inverted pendlum)
4. central pattern generators (centrálne vzorové generátory)

3.1.9.2.1 Ďalšie algoritmy:

- Genetic algoritmus
- Inverted pendlum(obratene kyvadlo)
- Hill climbing
- Cross-entropy method
- Covariance matrix adaptation evolution strategy

3.1.9.2.2 Genetic algoritmus

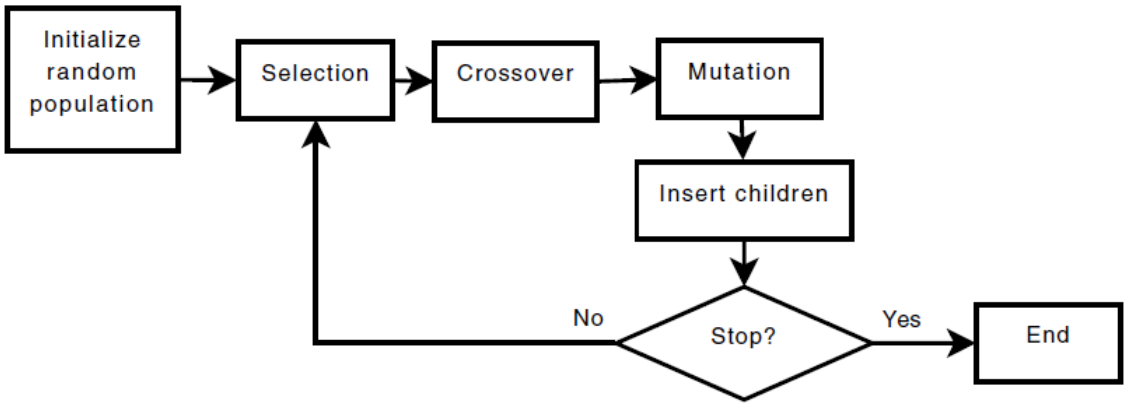


Schéma pre geneticky algoritmus

Selection – vyber rodičov pre kríženie podľa preddefinovaných pravidiel(cena funkcie/kondície)

Crossover – generovanie potomkov rodičov, výmenou niektorých génov pomocou niektorých schém: one-point, two-point, uniform(rovnomerne?). Potomkovia tak dedia niektoré vlastnosti od každého rodiča.

Mutation – generuje potomka tak, že náhodne mení jeden alebo niekoľko génov. To umožňuje hľadanie nových oblasti riešení, ktoré by neboli inak preskúmané. Mutácia preto zabraňuje GA

sústrediť sa len na lokálne vyhľadávanie, ktoré zase zvyšuje pravdepodobnosť nájdenia globálneho maxima.

Population ← *CreateInitialPopulation*()
Evaluate(*Population*)
while TerminationConditionNotMet() **do**
[Selection] *Parents* ← Selection(Population)
[Elistism] *Elite* ← Elitism(Population)
[Crossover] *Children* ← Crossover(*Parents*, *pc*)
[Mutation] *Mutants* ← Mutation(*Children*, *pm*)
Population ← *Elite* +*Mutants*
Evaluate(*Population*)
end while
return Best(*Population*)

3.1.9.2.3 Zdroje

- http://sabia.tic.udc.es/articulos/2009/BipedWalking.pdf

3.1.9.2.4 Cyklus chodenia

Majme robota s trupom a nohami. Každá noha pozostáva zo stehna, predkolenia, nohy. Majú 6 stupňov voľnosti. Tri v bedrovom kĺbe, jedna v kolene a dva v členku. Chodenie pozostáva z dvoch fáz. Buď sú obidve nohy na zemi alebo iba jedna, ktorá slúži ako opora. Počas fázy keď sú obide nohy na zemi sa postupne prenáša ťažisko zo zadnej nohy na prednú nohu. Ak je táto fáza pomalá, je náročné vytvoriť rýchlu chôdzu. Treba vedieť stanoviť správny čas na túto fázu.

Keď trajektórie pohybu bedrových kĺbov a nôh sú známe, vieme odvodiť uhly kĺbov pomocou kinematických obmedzení. Robot sa pohybuje rovno, polohy chodidiel bývajú konštante. Chôdza bude pozorovaná z boku.

Trajektória nohy popísaná $X_a = [x_a(t), z_a(t), \Theta_a(t)]^T$ kde $(x_a(t), z_a(t))$ sú súradnice členku a $\Theta_a(t)$ je uhol členku. Trajektória kĺbu popísaná vektorom $X_h = [x_h(t), z_h(t), \Theta_h(t)]^T$ kde $(x_h(t), z_h(t))$ sú súradnice bedrového kĺbu a $\Theta_h(t)$ je uhol bedrového kĺbu(Fig. 1).

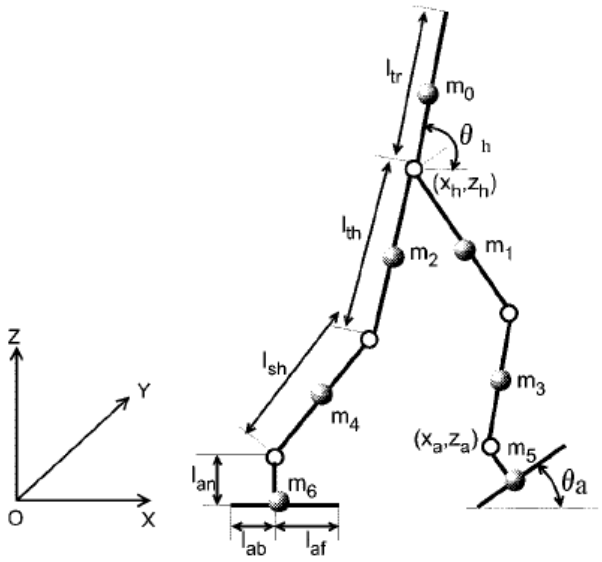


Fig. 1. Model of the biped robot.

3.1.9.2.5 Trajektória nôh

Časová perióda pre jeden cyklus chodenie je T_c. Je to čas jednej periódy cyklu chodenia z kT_c do $(k+1)T_c$. Kde k=1,2,...K, K je počet krokov. Na začiatku periódy sa päta pravej nohy dotýka zeme a odchádza, vtedy je v čase kT_c a perióda končí opäť keď sa pravá noha blíži k zemi a dotýka sa jej v čase $(k+1)T_c$, (Fig.2).

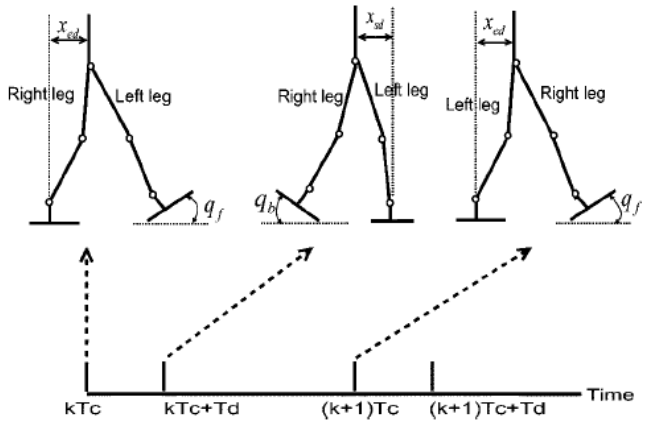


Fig. 2. Walking cycle.

Ďalej je popísané generovanie trajektórie pre pravú nohu. Trajektória ľavej nohy je rovnaká okrem trvania T_c.

Chôdza, pri ktorej sa robot dotýka celou nohou zeme hneď pri prvom kontakte nie je vhodná pre rýchlu chôdzu. Nech q_b a q_f sú uhly pravej nohy prichádzajúcej a odchádzajúcej na/zo zeme(fig. 2).

Predpokladajme, že pravá noha sa celá dotýka zeme v čase $t=kT_c$ a $t = (k+1)T_c + T_d$. Z toho dostávame

$$\theta_{\mathrm{a}}(t) = \begin{cases} q_{\mathrm{gs}}(k), & t = kT_{\mathrm{c}} \\ q_{\mathrm{b}}, & t = kT_{\mathrm{c}} + T_{\mathrm{d}} \\ -q_{\mathrm{f}}, & t = (k+1)T_{\mathrm{c}} \\ -q_{\mathrm{ge}}(k), & t = (k+1)T_{\mathrm{c}} + T_{\mathrm{d}} \end{cases} \quad (1)$$

- T_d je interval s oboma nohami na zemi
- $q_{gs}(k)$ a $q_{ge}(k)$ sú uhly medzi plochou a nohami

Na nerovnom teréne s prekážkami očakávame aby robot vedel prekážky prekonať. Nech (L_{ao}, H_{ao}) je pozícia kde sa noha vo vzduchu nachádza najvyššie(fig.3).

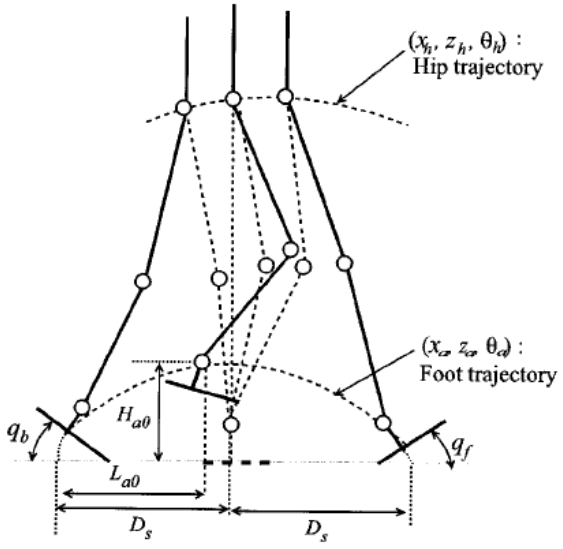


Fig. 3. Walking parameters.

$$x_{\mathrm{a}}(t) = \begin{cases} kD_{\mathrm{s}}, & t = kT_{\mathrm{c}} \\ kD_{\mathrm{s}} + l_{\mathrm{an}} \sin q_{\mathrm{b}} + l_{\mathrm{af}}(1 - \cos q_{\mathrm{b}}), & t = kT_{\mathrm{c}} + T_{\mathrm{d}} \\ kD_{\mathrm{s}} + L_{\mathrm{ao}}, & t = kT_{\mathrm{c}} + T_{\mathrm{m}} \\ (k+2)D_{\mathrm{s}} - l_{\mathrm{an}} \sin q_{\mathrm{f}} - l_{\mathrm{ab}}(1 - \cos q_{\mathrm{f}}), & t = (k+1)T_{\mathrm{c}} \\ (k+2)D_{\mathrm{s}}, & t = (k+1)T_{\mathrm{c}} + T_{\mathrm{d}} \end{cases} \quad (2)$$

$$z_{\mathrm{a}}(t) = \begin{cases} h_{\mathrm{gs}}(k) + l_{\mathrm{an}}, & t = kT_{\mathrm{c}} \\ h_{\mathrm{gs}}(k) + l_{\mathrm{af}} \sin q_{\mathrm{b}} + l_{\mathrm{an}} \cos q_{\mathrm{b}}, & t = kT_{\mathrm{c}} + T_{\mathrm{d}} \\ H_{\mathrm{ao}}, & t = kT_{\mathrm{c}} + T_{\mathrm{m}} \\ h_{\mathrm{ge}}(k) + l_{\mathrm{ab}} \sin q_{\mathrm{f}} + l_{\mathrm{an}} \cos q_{\mathrm{f}}, & t = (k+1)T_{\mathrm{c}} \\ h_{\mathrm{ge}}(k) + l_{\mathrm{an}}, & t = (k+1)T_{\mathrm{c}} + T_{\mathrm{d}} \end{cases} \quad (3)$$

- D_s je dĺžka kroku
- $kT_c + T_m$ ječas kedy je noha v najvyššej pozícii
- l_{an} je výška nohy k členku (fig.1)
- l_{af} je dĺžka z členku k prstom na nohách (fig. 1)
- l_{ab} je dĺžka z členku k päte (fig.1)
- $h_{gs}(k)$ a $h_{ge}(k)$ sú dĺžky povrchu, na ktorých stojí noha

Keď celý povrch pravej nohy je na zemi v čase $t=kT_c$ a $t=(k+1)T_c + T_d$ tak platia nasledujúce obmedzenia:

$$\begin{cases} \dot{\theta}_a(kT_c) = 0 \\ \dot{\theta}_a((k+1)T_c + T_d) = 0 \end{cases} \quad (4)$$

$$\begin{cases} \dot{x}_a(kT_c) = 0 \\ \dot{x}_a((k+1)T_c + T_d) = 0 \end{cases} \quad (5)$$

$$\begin{cases} \dot{z}_a(kT_c) = 0 \\ \dot{z}_a((k+1)T_c + T_d) = 0. \end{cases} \quad (6)$$

Na generovanie hladkej trajektórie je potrebné aby

- prvá derivácia(rýchlosť) $x_a(t)$, $z_a(t)$ a $\Theta_a(t)$ boli diferencovateľné
- druhá derivácia(zrýchlenie) $x_a(t)$, $z_a(t)$ a $\Theta_a(t)$ boli spojité na t, vrátane takých t, ktoré sú rovné kT_c, $kT_c + kT_d$, $kT_c + T_m$, $(k + 1)T_c$, $(k + 1)T_c + T_d$

Splniť obmedzenia (1) až (6) bude príliš drahé a výpočet pomocou polynomiálnej interpolácie je zložitý. Trajektóriu nôh získame z 3rd-order spline interpolácie. Druhé derivácie potom vždy budú spojité. Nastavením $q_{gs}(k)$, $g_{ge}(k)$, $h_{gs}(k)$, $h_{ge}(k)$, q_b, q_f, H_{ao}, L_{ao} sa dajú vytvoriť rôzne trajektórie nôh.

3.1.9.2.6 Trajektória bedrového kĺbu

Ak nie je žiadny driekový kĺb je želateľné aby $\Theta_h(t)$ bol konštantný. $\Theta_h(t)=0.5\ \Pi$ rad. Pohyb $z_h(t)$ ovplyvňuje ZMP(zero moment point). Predpokladajme, že kĺb je v najvyššej pozícii $H_{h\ max}$v strede fázy kedy je jedna noha na zemi a v najnižšej pozícii $H_{h\ min}$ vo fáze s dvoma nohami na zemi. Potom

$$z_h(t) = \begin{cases} H_{h\,min}, & t = kT_c + 0.5T_d \\ H_{h\,max}, & t = kT_c + 0.5(T_c - T_d) \\ H_{h\,min}, & t = (k+1)T_c + 0.5T_d. \end{cases} \quad (7)$$

Trajektória $z_h(t)$ splňujúca (7) a spojitosť po druhej derivácii sa získa z 3rd-order spline interpolácie.

Zmena $x_h(t)$ je hlavný faktor ovplyvňujúci stabilitu dvojnohého robota. Existujú metódy na odvodenie trajektórie bedrového kĺbu na vykonanie ZMP. Nie všetky trajektórie sú dosiahnuteľné a akcelerácia kĺbu musí byť veľká. Na vyriešenie konfliktov boli navrhnuté opatrenia

- generovanie série hladkých pohybov $x_h(t)$
- stanoviť hranicu $x_h(t)$

Proces chodenia pozostáva z troch fáz.

- Štartovacia fáza, v ktorej rýchlosť sa mení z 0 na želanú konštantnú rýchlosť.
- Ustálená fáza s želanou konštantnou rýchlosťou
- Koncová fáza, v ktorej rýchlosť sa mení z konštantnej rýchlosti na 0.

$X_h(t)$ ustálenej fázy je získaná z procedúry. $X_h(t)$ sa dá vyjadriť dvoma spôsobmi. Počas fázy s jednou nohou na zemi a fáza s dvoma nohami na zemi. X_{sd} a x_{ed} značia vzdialenosti k osi x z bedrového kĺbu k členku opornej nohy(fig.2).

$$x_{\mathrm{h}}(t) = \begin{cases} kD_{\mathrm{s}} + x_{\mathrm{ed}}, & t = kT_{\mathrm{c}} \\ (k+1)D_{\mathrm{s}} - x_{\mathrm{sd}}, & t = kT_{\mathrm{c}} + T_{\mathrm{d}} \\ (k+1)D_{\mathrm{s}} + x_{\mathrm{ed}}, & t = (k+1)T_{\mathrm{c}}. \end{cases} \tag{8}$$

Na získanie hladkého pohybu $x_h(t)$ v ustálenej fáze musia platiť obmedzenia s deriváciami.

$$\begin{cases} \dot{x}_{\mathrm{h}}(kT_{\mathrm{c}}) = \dot{x}_{\mathrm{h}}(kT_{\mathrm{c}} + T_{\mathrm{c}}) \\ \ddot{x}_{\mathrm{h}}(kT_{\mathrm{c}}) = \ddot{x}_{\mathrm{h}}(kT_{\mathrm{c}} + T_{\mathrm{c}}). \end{cases} \tag{9}$$

$$x_{\mathrm{h}}(t) = \begin{cases} kD_{\mathrm{s}} + \dfrac{D_{\mathrm{s}} - x_{\mathrm{ed}} - x_{\mathrm{sd}}}{T_{\mathrm{d}}^2(T_{\mathrm{c}} - T_{\mathrm{d}})}\big[(T_{\mathrm{d}} + kT_{\mathrm{c}} - t)^3 - (t - kT_{\mathrm{c}})^3 \\ \quad -T_{\mathrm{d}}^2(T_{\mathrm{d}} + kT_{\mathrm{c}} - t) + T_{\mathrm{d}}^2(t - kT_{\mathrm{c}})\big] + \dfrac{x_{\mathrm{ed}}}{T_{\mathrm{d}}}(T_{\mathrm{d}} + kT_{\mathrm{c}} - t) + \dfrac{D_{\mathrm{s}} - x_{\mathrm{sd}}}{T_{\mathrm{d}}}(t - kT_{\mathrm{c}}), & t \in (kT_{\mathrm{c}}, kT_{\mathrm{c}} + T_{\mathrm{d}}) \\ kD_{\mathrm{s}} + \dfrac{D_{\mathrm{s}} - x_{\mathrm{ed}} - x_{\mathrm{sd}}}{T_{\mathrm{d}}(T_{\mathrm{c}} - T_{\mathrm{d}})^2}\big[(t - kT_{\mathrm{c}} - T_{\mathrm{d}})^3 - (T_{\mathrm{c}} + kT_{\mathrm{c}} - t)^3 - (T_{\mathrm{c}} - T_{\mathrm{d}})^2(T_{\mathrm{c}} + kT_{\mathrm{c}} - t) \\ \quad -(T_{\mathrm{c}} - T_{\mathrm{d}})^2(t - kT_{\mathrm{c}} - T_{\mathrm{d}})\big] + \dfrac{D_{\mathrm{s}} - x_{\mathrm{sd}}}{T_{\mathrm{c}} - T_{\mathrm{d}}}(T_{\mathrm{c}} + kT_{\mathrm{c}} - t) + \dfrac{D_{\mathrm{s}} + x_{\mathrm{ed}}}{T_{\mathrm{c}} - T_{\mathrm{d}}}(t - kT_{\mathrm{c}} - T_{\mathrm{d}}), & t \in (kT_{\mathrm{c}} + T_{\mathrm{d}}, kT_{\mathrm{c}} + T_{\mathrm{c}}). \end{cases} \tag{10}$$

Pomocou 3rd order periodic spline interpolácie získame $x_h(t)$ splňajúce (8), (9) a podmienku spojitosti druhej derivácie danej (10). Určením rôznych x_{sd} a x_{ed} dostávame rôzne plynulé trajektórie $x_h(t)$ podľa (10).

$$\begin{cases} 0.0 < x_{\mathrm{sd}} < 0.5D_{\mathrm{s}} \\ 0.0 < x_{\mathrm{ed}} 0.5D_{\mathrm{s}}. \end{cases} \tag{11}$$

Na základe (10), (11), (13), (14) je odvodená hranica stability

$$\max_{x_{\mathrm{ed}} \in (0, 0.5D_{\mathrm{s}}), x_{\mathrm{sd}} \in (0, 0.5D_{\mathrm{s}})} d_{\mathrm{zmp}}(x_{\mathrm{sd}}, x_{\mathrm{ed}}) \tag{12}$$

$d_{zmp}(x_{sd}, x_{ed})$ určujú hranice stability. Sú iba dva parametre takže ľahko získame riešenie (12) pomocou vyčerpávajúceho hľadania(fig.4).

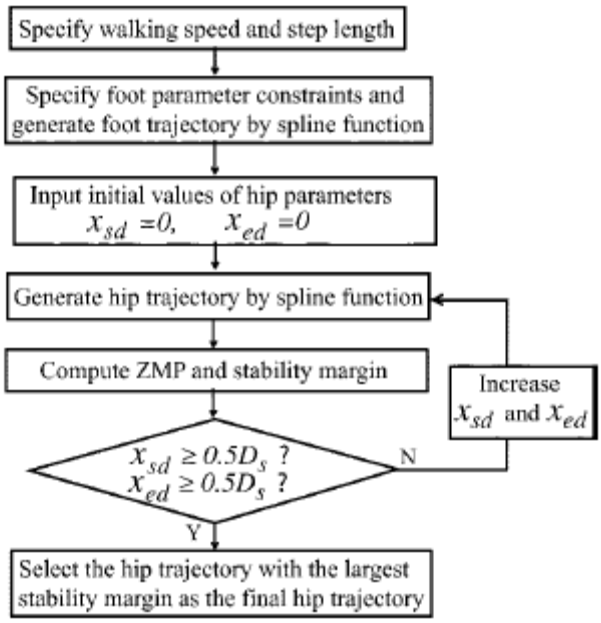


Fig. 4. Algorithm for planning walking patterns.

Určením $x_h(t)$ v ustálenej fáze sa špecifikujú koncové obmedzenia pre štartovaciu fázu a počiatočné obmedzenia pre koncovú fázu. Počiatočné obmedzenie pre štartovaciu fázu je aby derivácie $x_h(t_0)=0$ a koncové obmedzenia pre finálnu fázu $x_h(t_c)=0$. $X_h(t)$ štartovacej a koncovej fázy sa dajú získať 3rd order spline interpoláciou.

$$x_{\text{zmp}} = \frac{\sum_{i=1}^{n} m_i(\ddot{z}_i + g)x_i - \sum_{i=1}^{n} m_i\ddot{x}_i z_i - \sum_{i=1}^{n} I_{iy}\ddot{\Omega}_{iy}}{\sum_{i=1}^{n} m_i(\ddot{z}_i + g)} \tag{13}$$

$$y_{\text{zmp}} = \frac{\sum_{i=1}^{n} m_i(\ddot{z}_i + g)y_i - \sum_{i=1}^{n} m_i\ddot{y}_i z_i - \sum_{i=1}^{n} I_{ix}\ddot{\Omega}_{ix}}{\sum_{i=1}^{n} m_i(\ddot{z}_i + g)} \tag{14}$$

3.1.9.2.7 Zdroje

- http://staff.aist.go.jp/k.kaneko/publications/2001_publications/6.pdf

3.1.10 Git (pp 1.10, pp 1.11)

3.1.10.1 Analýza

Pre verziovanie softvérového produktu sme sa rozhodli použiť git hlavne pre výhodu necentralizovaného repozitára. Možnosti spoločného repozitára boli vytvorenie repozitára na našom serveri alebo použiť externú službu ako napríklad gitbus.fiit.stuba.sk.

3.1.10.2 Návrh

Náš tím použil externú slúžbu repozitára gitbus.fiit.stuba.sk. Kde sme vytvorili tím a k tímu sme vytvorili projekt, kde sme mali náš spoločný repozitár. Každý člen tímu si musel naištalovať na svojom stroji git, následne si vygenerovať rsa klúč pre autentifikáciu so spoločným git repozitárom, ktorý si uložili v službe gitbus.fiit.stuba.sk pod svojim kontom. Nakoniec si každý člen stiahol obraz repozitáru na svoj stroj a nastavil si vzdialený pôvodný repozitár na alias origin.

3.1.10.3 Implementácia

Pri implementácii každý člen tímu postupoval návodom inštalácie a použitia gitu pre tímový projekt, ktorý je v prílohe dokumentácie. S gitom každý člen postupoval podľa metodiky verziovacich systémov.

3.1.11 Manažérsky softvér (pp 1.12)

3.1.11.1 Analýza

Analýza prebehla porovnávaním rôznych manažérskych nástrojov, ktoré spĺňali základné kritériá pre náš projekt a pre distribuovanú tímovú prácu. Kritériá:

- Vzdialený prístup z webu
- Vytváranie backlogu
- Možnosť generovania Burndown chartu
- Možnosť generovania Ganttovho diagramu

3.1.11.1.1 Redmine

Redmine je voľne dostupný open source manažérsky nástroj. Jeho výhodou je ľahká rozširovateľnosť ďalšími modulmi. Obsahuje súborový manažment a manažment dokumentov. Ponúka e-mailovú notifikáciu. Je vhodná na agilný vývoj softvéru.

3.1.11.1.2 Jira

Je komerčný program na bug-tracking a vývojový manažment. Implementovaný je v Jave. Jej používanie je veľmi ľahké. Hodí sa na agilný vývoj softvéru. Obsahuje spárovanie zdrojových kódov s úlohami.

3.1.11.1.3 dotProject

Implementovaný je v PHP spolu s MySQL. Je opensource a free softvér. Podporuje použitie Ganttovho diagramu. Obsahuje generovanie reportov aj diskusné fórum.

3.1.11.2 Výstup analýzy

Redmine nám vyhovoal najviac zo všetkých možností. Okrem opísaných výhod obsahoval všetky žiadané funkcie. Jeho prístup cez školský server bol veľmi ľahký.

3.2 Šprint 2 – „beta“

Tabuľka 3.4: Príbehy v šprint backlogu

ID pp	Ako	Chcem	Aby bolo možné
2.1	Člen tímu	Stanoviť celkovú prácu počas projektu	Vyberať úlohy, na ktorých sa bude pracovať
2.2	Člen tímu	Importovať a skrížiť hráčov minuloročných projektov	Pracovať už na funkčnom a spojenom projekte
2.3	Člen tímu	Porovnať a zanalyzovať zdrojové kódy minuloročných projektov	Zistiť v akom sú stave a čo je alebo nie je funkčné
2.4	Člen tímu	Vytvoriť dokumentáciu projektu	Celkovú prácu zaznamenať na jedno miesto a aby bola možná tlač
2.5	Člen tímu	Analyzovať chôdzu skríženého hráča	Aby bolo možné začať prvé zdokonaľovanie prípadne vyvíjanie
2.6	Člen tímu	Vytvoriť koding guide	Jednoznačne písať a komentovať zdrojový kód
2.7	Člen tímu	Vytvoriť metodiku na používanie Gitu	Bezproblémovo verziovať zdrojový kód

Tabuľka 3.5: Detailný opis úloh

ID pp	Zodpovedný pp	ID úlohy	Úloha	Zodpovedný
2.1	-	2.1	Návrh celkovej práce počas projektu	M. Červeňák
2.2	-	2.2	Git – importovanie a kríženie hráčov	M. Ondrejkovič
2.3	-	2.3	Analýza z. kódu hráča tímov High5 a Tím 17 žije...	M. Gregor
2.4	-	2.4	Vytvoriť dokumentáciu aj so šablónou	M. Červeňák
2.5	-	2.5	Analýza chôdze nášho (kríženého) hráča	G. Nagy
2.6	-	2.6	Koding guide (šablóna-metodika)	F. Sucháč
2.7	-	2.7	Metodika na používanie Git-u	M. Ondrejkovič

3.2.1 Návrh celkovej práce počas projektu (pp 2.1)

3.2.1.1 Analýza

Celý predchádzajúci šprint sa zaoberal inštaláciou komponentov na spustenie projektu a analýzou problematiky. A práve tá analýza bola dôležitá pre návrh celkovej časti na projekte tohto projektu. Jednotlivý členovia tímu sa oboznámili s dokumentáciami a wikipediou z minulého roka a analyzovali tímy úspešné v celosvetovej súťaži RoboCup. Medzi analyzované časti bola zahrnutá aj analýza fyzických modelov práve kvôli pochopeniu stability a pod. Predbežná analýza v tejto úlohe vyústila do spísanie základného návrhu práce, ktorým by sa bolo možné uberať.

3.2.1.2 Návrh

Na základe analýzy, ktorou sa tím zaoberal boli navrhnuté rôzne nápady, ktoré by bolo možné realizovať. Jednotlivé nápady a návrhy boli spracované do tabuľky, ku ktorým bol pridaný detailný opis. Následne po komunikácii a dohode sa z navrhnutých nápadov zostavil zoznam používateľských príbehov, ktoré sa zoradením priorít budú postupne vykonávať. Takýto priorizovaný zoznam používateľských príbehov tvorí produktový backlog projektu. Ten slúži ako základný stavebný prvok pri vývoji metodikou Scrum, ktorou sa uberá aj tento projekt.

Výstupom tejto úlohy je spísanie návrhov jednotlivých členov tímu, ktoré sa nachádzajú v tabuľke 3.6. Pod tabuľkou sú detailné opisy niektorých úloh.

Tabuľka 3.6: Navrhnuté úlohy

Návrh	Popis	Navrhol
Optimalizácia vytvorených nízkoúrovňových pohybov	• dosiahnuť kvalitu vhodnú na použitie pri vytváraní vysokoúrovňových pohybov (stabilita, rýchlosť). • testovať ich, a to najmä pomocou gyroskopu a akcelometra, ktoré obsahuje testovací framework. • vylepšenie high skillu kráčania za loptou - aby hráč vedel loptu viesť • vylepšenie úkrokov - aby sa hráč nemusel dlho otáčať a natáčať • rýchlosť základných pohybov • kop do lopty	J. Gregor M. Červeňák G. Nagy F. Sucháč
Vytvorenie nových pohybov	• beh • otáčanie hráča počas chôdze resp. chôdza po kružniciach, aby nemusel zastavovať a natáčať sa. • viac pohybov pri bránení • ochrana lopty telom	J. Gregor M. Červeňák M. Ondrejkovič
Vytvorenie univerzálneho trénera	• zozbieranie všetkých informácii z ihriska (pozícia lopty, spoluhráčov, protihráčovov) • poskytovanie informácie svojim hráčom • vyhodnotenie získaných informácií a určenie, čo majú hráči robiť (ako sa majú rozostaviť, kto ma isť za loptou, akú stratégiu majú voliť...)	J. Gregor
Nástroj pre trénovanie stratégií	• nastavovanie, teda rozloženie hráčov • zmena správania hráča	M. Ondrejkovič

Vylepšovanie editora pohybu	• identifikovanie fázy, kedy robot už stráca stabilitu • zistenie ako by sa malo zmeniť ťažisko robota alebo čo spraviť aby sa udržala stabilita v danej fáze pohybu • napovedanie ako by sa mali meniť uhly kĺbov a časy ich ohýbania, príp. algoritmus, ktorý by skúšal rôzne riešenia	M. Ondrejkovič G. Nagy
	• vylepšiť, prepracovať, poprípade napísať úplne nový editor pohybov	
Experimenty zadefinovania polôch	• dlhodobý záznam stavov kĺbov a snímačov v situáciách, kde sa stráca rovnováha a dochádza k neúmyselnému pádu • navrhnutie nového stabilizačného mechanizmu pomocou nazbieraných údajov	G. Nagy
Roboviz - vylepšenie testovacieho frameworku	• vylepšenie nášho testovacieho frameworku podľa RoboVizu • implementovanie aspoň jednej vlastnosti RoboVizu do nášho testovacieho frameworku.	M. Gregor
Positioning to Win - použitie formácií pri robotickom futbale	• náročné na implementáciu lebo vyžaduje funkčné a stabilné nízkoúrovňové pohyby • táto časť návrhu závisí od dokonale zvládnutej časti základných pohybov hráča	M. Gregor

Nástroj pre trénovanie stratégií

Vytvorenie nástroja pre trénovanie stratégií. Hlavnou častou je nastavovanie, teda rozloženie hráčov, zmena správania hráča. Tento nastroj by mal hlavne význam napr. pre analýzu danej situácie za rôznych predpokladov (priamy kop, nahrávky do priestoru, útok) alebo rôznych situácií pri rovnakej stratégii. Výhodne aj pre overovanie a testovanie stratégií.

Vylepšovanie editora pohybu

Návrh 1

Editor umožňuje vytvárať pohyby ale vytvorený pohyb nezaručuje, že pohyb robota bude stabilný a udrží sa na nohách, preto vytvoriť metódy pre vytváranie stabilných pohybov:

- identifikovanie fázy, kedy robot už stráca stabilitu(sa potáca)

- zistenie ako by sa malo zmeniť ťažisko robota alebo čo spraviť aby sa udržala stabilita v danej fáze pohybu
- napovedanie ako by sa mali meniť uhly kĺbov a časy ich ohýbania, príp. algoritmus, ktorý by skúšal rôzne riešenia prípadne nový prvok – ochrana lopty telom.

Návrh 2

Vhodné by bolo vylepšiť, prepracovať, poprípade napísať úplne nový editor pohybov. Editor by mohol pracovať aj na úplne inom princípe. Cieľom vylepšeného editora by nebolo obohatenie funkcií, skôr vylepšenie základných funkcií z dôvodu, že editor používajú začiatočníci. Pomohlo by to k rýchlejsiemu pochopeniu pohybu robota pre začiatočníkov, aby si osvojili rýchlejšie postup vytvárania pohybov, získali skúsenosť a takto sa rýchlejšie dostali na úroveň, keď už písaním XML súborov vytvárajú kvalitné pohyby.

Návrh na postup práce:

- Otestovať funkcie súčasného editora
- Otestovať kompatibilitu s operačnými systémami
- Navrhnúť nový koncept vytvárania pohybov, alebo navrhnúť zmeny súčasného riešenia
- Implementovať program
- Nový editor sprístupniť pomocou wiki RoboCupu na FIIT

Experimenty zadefinovania polôh

Dlhým testovaním by sa zadefinovali kombinácie natočenia kĺbov a stavov snímačov, pri ktorých dochádza k strate rovnováhy. Použitím výsledkov by sa dal vytvoriť kontrolný mechanizmus, ktorý by sledoval stav robota a zabraňoval by pádu a straty rovnováhy.
Návrh riešenia:

- Dlhodobý záznam stavov kĺbov a snímačov v situáciách, kde sa stáca rovnováha a dochádza k neúmyselnému pádu
- Navrhnúť nové stabilizačné mechanizmy pomocou nazbieraných údajov
- Implementovať riešenie

Roboviz - vylepšenie testovacieho frameworku

Roboviz je program na monitorovanie a podporu vývoja agenta v multiagentovom prostredí. Jeho dokumentácia, návod na inštaláciu, ukážky použitia a návrhy ďalšieho vylepšenia sú na adrese https://sites.google.com/site/umroboviz/home. Je to náhrada Simsparkového monitora robotického futbalu, ktorý navyše vykresľuje agenta s informáciami o stave sveta v 3D prostredí. Roboviz ponúka funkcionality ako programovateľné vykresľovanie rôznych útvarov do ihriska, ktoré poslúžia napríklad pri testovaní obhádzania objektov agentom. Prináša aj možnosť debugovania komunikácie agentov prostredníctvom siete. Medzi hlavné vlastnosti patrí:

- v dvojrozmernom grafickom podaní zobrazuje pod hráčom vektory smeru, kde sa agent pozerá, kde počúva a hovorí alebo aj akým smerom chce napredovať. Tiež zobrazuje kde hráč vidí loptu. Táto funkcionalita je prospešná pri dolaďovaní pohybových a lokalizačných algoritmoch.
- prostredníctvom Robovizu sa dá modelovať svet. Dajú sa napríklad pohybovať hráči, lopta, hráči sa môžu pridávať alebo odoberať.
- už spomínané kreslenie tvarov pre testovanie alebo vývoj algoritmov vyhýbania sa predmetom respektíve iným hráčom. Dá sa to využiť aj pre učenie sa nových formácií prostredníctvom strojového učenia sa a nie programovaním schopností.
- potom už len zostávajú vlastnosti ako možnosť nastavovať alebo reštartovať server, lepšia grafika zobrazovania hry, menu alebo možnosť vidieť svet z perspektívy hráča

Nevýhodu z technického hľadiska, ktorú vidíme je potreba simsparku skompilovaného bez multithreadingu, ktorého vypnutie môže spôsobiť nesprávny alebo pomalší chod servera. Náš testovací framework vylepšíme implementovaním vlastností ako sú v robovize. Pre hlbšiu anlýzu Robovizu sa dostupné zdrojové súbory.

Positioning to Win - použitie formácií pri robotickom futbale

Tím robotického futbalu Austin Villa vyvinuli pozičný a formačný systém, ktorý im dopomohol k výhre viacerých zápasov. Ich pozičný systém rozhoduje v troch krokoch:

1. Najlepšia formácia sa vypočíta podľa algoritmu prideľovania pozícií pre celý tím distribuovaním si pozícii medzi hráčmi navzájom.

2. Najlepšiu formáciu si vypočíta každý hráč podľa toho čo vidí.

3. Zo všetkých vypočítaných formácií sa vyberie tá najlepšia pre súčasnú situáciu.

Sťažujúcimi okolnosťami, s ktorými treba počítať pri výbere formácie sú obmedzenosť komunikácie a toho čo hráč vidí, motorickosť hráčov a zvládnutie nízkoúrovňových pohybov. Formácie, ktoré používa tím Austin Villa sú hlavne obranná formácia a útočná formácia. Prideľovanie pozícií formácie vychádza z pravidiel, že brankár sa do formácie nezapája lebo bráni bránkovisko. Najbližší hráč k lopte sa do formácie nezapája, ale urýchlene sa snaží dostať k lopte a nahrať ju spoluhráčovi alebo vystreliť. Ostatní hráči sa snažia dostať na určené pozície. Náš hráč po zvládnutí nízkoúrovňových pohybov bude ovládať formáciu podľa opísaných pravidiel. Formáciu sa naučí dynamickým spôsobom (strojovým učením sa) alebo naprogramovaním formácie.

3.2.2 Import a kríženie hráčov (pp 2.2)

3.2.2.1 Analýza

Zdrojové kódy je treba najprv analyzovať, čo z časti pokryje task 3.2.3. Agenti dvoch tímov sú veľmi podobní, z dôvodu, že tímy podstatnú časť projektu vyvíjali spolu. Test Framework školského projektu sa líšil iba v testovacích triedach.

3.2.2.2 Návrh

Porovnávanie zdrojových kódov prebehne vo vývojovom nástroji alebo textovom editore. Po zistení nezhody sa kódy krížia na základe niektorého z uvedených princípov:

- Použije sa funkčný kód
- Použije sa lepší kód (kvalitnejší, alebo s lepším výstupom)
- Pomocou dvoch riešení sa vytvorí tretie riešenie

Vyhodnotenie kódov prebehne testovaním, debugovaním alebo logickou simuláciou. V prípade, že sa kódy nevedia dostatočne otestovať aby sa zvolil jeden z princípov, tak do kríženého hráča doplníme oba kódy tým, že sa jeden z nich zakomentuje a posunie sa na neskoršie testovanie.

3.2.3 Analýza zdrojových kódov z minulého roka (pp 2.3)

3.2.3.1 Analýza

Na analýzu doterajších vlastností školského projektu sme si zobrali minuloročné projekty. Projekt sme analyzovali v 2 častiach a to Jim a TestFramework. Školský projekt obsahuje aj tretiu časť RoboCupLibrary, ale táto časť sa postupne prestáva implementovať v novších verziách produktu školského projektu. Preto analýza tretej časti sa dotýkala len implementovaných prvkov. Jim bol dosť rozdielny medzi tímom 5 a tímom 17 z minulého roku lebo skúšali rôzne prístupy. Testovací framework sa líšil iba vo vektorových nastaveniach v naprogramovaných testoch. Bolo potrebné zistiť funkčnosť jednotlivých častí.

3.2.3.2 Návrh

Každý člen tímu si zobral na starosť tretinu z množstva 2 projektov Jim a TestFrameworku. Funkcionalita jednotlivých funkčných prvkov sa testovala použitím Simspark servera a monitora zapínaním/vypínaním alebo modifikovaním nastavení jednotlivých prvkov. Výsledok analýzy každý člen zhrnul do preddefinovanej tabuľky, ktorá bude súčasťou zápisnice z 6. Stretnutia. Z analýzy vyplynuli nasledujúce funkčné funkcie:

- sk.fiit.jim.Settings – funguje globálne nastavenie hry
- sk.fiit.agent.AgentInfo – funguje a vracia informácie o agentovi, jeho polohe , polohe vzhľadom na loptu
- sk.fiit.agentcommunication – funguje. Zabezpečuje komunikáciu zo serverom, posielanie aj prijímanie správ.
- sk.fiit.agent.models.AgentModel – obsahuje model agenta. Údaje o polohe, natočení a výpočty z accelerometra.
- sk.fiit.agent.models.AgentPositionCalculator – počíta polohu agenta na základe vlajok, ktoré vidí.
- sk.fiit.agent.models.AgentRotationCalculator – počíta natočenie agenta podľa vlajok, ktoré vidí.

- sk.fiit.agent.models.DynamicObject – vypočíta polohu pohybujúceho sa objektu ako napríklad lopty.
- sk.fiit.agent.models.EnvironmentModel – model sveta hry, ktorý obsahuje informácie ako čas hry, stav hry, verzia servera.
- sk.fiit.agent.models.TacticalInfo – obsahuje informáciu o hernej situácii (útočíme, bránime, ...)
- sk.fiit.agent.moves – balík obsahuje triedy, ktoré definujú rozsah natočiteľnosti kĺbov, algoritmus opravy chýb natočenia kĺbov, triedy spracovania lowsklillov. Všetko funguje.
- sk.fiit.jim.log – balík obsahuje triedy na logovanie akcií v simulovanom robotickom futbale.
- sk.fiit.jim.init – balík svojimi triedami zabezpečuje inicializáciu hráča na server, čo znamená zabezpečiť vloženie agenta, načítanie ruby skriptov a ich otestovanie, načítanie xml súborov s pohybmi pre agenta
- sk.fiit.jim.gui – balík s triedami grafického rozhrania na znovu načítanie xml súborov s pohybmi aby sa nemusel reštartovať server
- sk.fiit.jim.build – balík s triedami, ktoré zabezpečujú vytvorenie a testovanie release súboru (balíka) s ruby skriptami, xml súbormi a potrebnými java knižnicami.
- sk.fiit.jim.annotation.gui – balík obsahuje triedy na rozpoznanie a načítanie xml súborov.
- sk.fiit.jim.agent.parsing – parsovanie správ a prijímanie zvukových a vizuálnych správ.
- sk.fiit.jim.agent.server – obsahuje tiredu na multi vláknový server.
- sk.fiit.agent.skills – pokus o riešenie high skillov, funguje, ale nie je to dokonalé riešenie. Je tam čo vylepšovať.
- sk.fiit.jim.annotation – správa vytvárania pohybov z anotácií. Funguje len manažér anotácii a určenie osí sveta.
- sk.fiit.testframework.init – balík, ktorý obsahuje triedy na spustenie a nastavenie testframeworku. Je možné si definovať rôzne implementácie spustenia frameworku ako napríklad na viacerých vláknach. Funguje všetko až na MpiImplementation.
- sk.fiit.testframework.monitor – balík obsahuje triedy, ktoré definujú typy správ, ktoré sa na serveri posielajú, triedy ktoré pridávajú a odoberajú vlákna agentov a monitorov a zabezpečujú pridávanie listenerov na agentov.
- sk.fiit.testframework.communication.agent – balík obsahuje triedy spracúvajúce komunikáciu medzi agentmi, ale tiež udržujú aj informácie o hráčovi ako tím, číslo dresu, pridelenie voľného tftp portu, pridanie a odobratie agenta alebo agentov. Tiež aj triedy predstavujúce model Agenta pridaného serverom a alebo agenta Jima.
- sk.fiit.testframework.communication.robocupserver – balík, ktorý obsahuje triedy na modifikáciu situácie na ihrisku. Dokážeme vykonať príkaz na serveri, posunúť hráča, loptu.
- sk.fiit.testframework.beta.ui – balík reprezentujúci grafické rozhranie testframeworku. Rozhranie je generované pomocou Jigloo.
- sk.fiit.testframework.annotator.serialization – balík na vytváranie pohybov na základe anotácii zoserializuje viacero nízkoúrovňových pohybov do jedného xml.
- sk.fiit.testframework.agenttrainer.models – triedy zodpovedné za synchronizáciu fáz pohybu, gettery a settery efektorov. Využíva sa aj pri spúšťaní pohybov z xml a to funguje.
- sk.fiit.testframework.worldPresentation – interpratácia modelu sveta a prenesenie správ do sveta, reprezentácie hráča a nastavovanie pozícií agentov

- sk.fiit.testframework.parsing - parsovanie správ, typy správ, grafy. Funguje.

Nefungujúce časti kódu:

- sk.fiit.testframework.agenttrainer – balík, ktorý má zabezpečiť trénera nových pohybov, ale neobsahuje nijaký algoritmus, ktorý by to riešil.
- sk.fiit.testframework.communication.agent.AgentJim – nefunguje reštart servera
- sk.fiit.testframework.annotator.AnnotatorTestCaseResult – trieda má zabezpečiť výpis správ pri testovaní pohybov, ale nič nevypisovala ani pri zmene granularity výpisu správ.

3.2.4 Tvorba dokumentácie (pp 2.4)

3.2.4.1 Analýza

Tímová dokumentácia pozostáva z dvoch typov dokumentácie:

- Dokumentácie k inžinierskemu dielu
- Dokumentácie k riadeniu

Obsahom prvej z nich je technická dokumentácia celého projektu a jednotlivých analyzovaných či implementačných úloh.

Obsahom druhej dokumentácie je riadenie projektu. Pozostáva z častí, ktoré sa týkajú riadenia ľudí, vývoja a pod.

3.2.4.2 Návrh

Obsahy oboch dokumentácií vznikali od začiatku projektu v podobe metodík a analýzy rôznych častí a aspektov pri vývoji. Neskôr boli vytvorené 2 veľké dokumenty, ktoré tieto dokumentácie zahrnuli. Tieto štruktúrované dokumenty sú zhromaždením všetkých informácii o projekte a samotnom riadení projektu. Dokumentácia je pravidelne dopĺňaná a aktualizovaná.

3.2.5 Návod na písanie kódu (pp 2.6)

3.2.5.1 Analýza

Analýza pre návod na písanie kódu pozostávala z analýzy existujúcich zdrojových kódov pre RoboCup a z čítania knihy Martin, R.C.: Clean Code – A Handbook of Agile Software Craftsmanship. Pearson Education, Inc. 2009. ISBN 0-13-235088-2.

3.2.5.2 Návrh

3.2.5.2.1 JavaDoc komentáre pre triedy a metódy

```
/**
 *  Settings.java
 *
```

```
 *  Encapsulates global settings that alter the behaviour of the code throughout
 *  the entire project. Can change default settings for the game. Usually is
 *  meant to be set in ./scripts/config/settings.rb.
 *
 *  @Title    Jim
 *  @author   marosurbanec
 *  @author   Androids
 */
public final class Settings { ...
```

Nad každou triedou je potrebné vytvoriť JavaDoc komentár v minimálne takejto forme:

- Názov súboru triedy
- Popis triedy (cca. 3 riadky)
- Titulok je názov projektu (Jim, TestFramework, ...)
- Autor vo formáte ako v AIS, napr: @author xsuchac
- Znovu autor, ale už s názvom tímu, teda @author A55-Kickers

```
/**
 * Checks employees and find out who is on vacation.
 *
 *  @author xsuchac
 *  @author A55-Kickers
 *
 * @param employeesToCheck – employees needed to be checked.
 * @param employeesOnVacation – employees who are on vacation.
 * @return resultEmployees – employees on vacation filtered out
 * of employeesToCheck.
 * @throws IllegalArgumentException – if any parameter is null.
 */
private static ArrayList getCheckedEmployeesOnVacation(ArrayList
employeesToCheck, ArrayList employeesOnVacation) throws
IllegalArgumentException { ...
```

Nad každou metódou:

- Popis metódy
- Autor vo formáte ako v AIS, napr: @author xsuchac
- Znovu autor, ale už s názvom tímu, teda @author A55-Kickers
- Parametre metódy (ak sú): @param názovParametra – popis parametra
- Návratová hodnota metódy: @return názovNávratovejHodnoty – popis
- Výnimky, ktoré môže metóda vyhadzovať (ak sú): @throws NázovVýnimky - popis

3.2.5.2.2 Komentáre všeobecne

Pri komentovaní metód a premenných je potrebné uvádzať nad ich definíciou JavaDoc komentár. Teda preferovať:

```
/** something */
```

pred

```
/* something */
```

Ukážka:

```
/**
 * Team of the agent
 */
public static String team;
```

Ostatné komentáre, ktoré sa neviažu na definíciu premennej, metódy, triedy a pod. môžu byť v základnom tvare `// something` alebo `/* something */`

Komentáre však používať iba v nutných prípadoch. Čisto svoj kód je potrebné radšej vytvoriť tak, aby dával zmysel aj ostatným (zmysluplnými názvami premenných, metód a pod.), namiesto aby sa len okomentoval.

3.2.5.2.3 Poznámky do kódu, zakomentovaný kód

Poznámky do kódu používať len pre osobnú krátkodobú potrebu. Pri odovzdávaní kódu do repozitára je potrebné ich zmazať alebo predtým príčinu poznámky vyriešiť. Kód sa pod vplyvom mnohých zmien a mnohých meniteľov kódu môže zmeniť tak, že poznámka úplne stratí zmysel a po čase si nikto nespomenie na to, čo znamenala a nikto ju pre istotu nebude chcieť vymazať.

Toto isté platí aj pre zakomentovanú časť kódu.

3.2.5.2.4 TODO komentár

V prípade, že si autor kódu v určitej časti kódu myslí, že by tam bolo potrebné niečo konkrétne dorobiť, ale z nejakého dôvodu to nemôže spraviť hneď, tak môže použiť TODO komentár. K týmto komentárom je potrebné sa pravidelne vracať a dať si záležať, aby bola funkcionalita na ktorú poukazujú doplnená a komentár odstránený. Výhodou TODO komentárov je, že je ich možné ľahko vyhľadať cez vývojové prostredie.

Obsah TODO komentára bude pozostávať z popisu a mena autora (tvar ako v AIS) s názvom tímu:

```
/* TODO Add special move #2 after discussion with colleagues.
 * @author xsuchac, A55-Kickers
 */
```

TODO komentár neberie do úvahy značky začínajúce zavináčom, preto môžeme použiť vyššie uvedenú skrátenú formu zápisu autora a tímu do TODO komentára.

3.2.5.2.5 Zmeny v kóde

Niektoré tímy si v kóde komentármi označujú časti kódu, ktoré zmenili. Túto praktiku robiť nebudeme. V prípade, že budeme potrebovať od predchádzajúcich autorov vysvetlenie k časti kódu, jej autorstvo zistíme podľa originál zdrojových kódov, ktoré sme od nich dostali.

3.2.5.2.6 Štruktúra triedy

Trieda by mala začínať zoznamom premenných. Public static konštanty by mali ísť najprv. Potom private static premenné, ďalej private atribúty inštancií. Len zriedkavo je dobré mať aj public premenné.

Public metódy by mali nasledovať za zoznamom premenných. Private metódy prislúchajúce k public metódam by mali ísť pod ne v poradí ako ich public metóda využíva. Toto pomáha čítať si triedu podobne ako novinový článok.

„Public“ dávať jedine premenným a metódam, ktoré budú využívať iné triedy!

3.2.5.2.7 Konvencia kódu

Používať klasickú java konvenciu. Ukážka:

```
public class ClassA {

	public int number = 1;
	private String letters[] = new String[]{“A”, “B”};

	public int meth(String text, int number) {
		if (text == null) {
			text = “a”;
		} else if (text.length() == 0) {
			text = “empty“;
		} else {
			number++;
		}
	}
}
```

Táto konvencia je defaultne nastavená v NetBeans a Eclipse. Pokiaľ ju nedodržiavate, tak je potrebné kód naformátovať:

- postup pre NetBeans:
  - pravým tlačidlom myši kliknúť do okna s kódom
  - vybrať „Format“
- postup pre Eclipse:
  - pravým tlačidlom myši kliknúť do okna s kódom
  - vybrať „Source“, kliknúť na „Format“

3.2.5.2.8 Zmysluplné názvy

Radšej použiť dlhší názov premennej, metódy a pod. ako taký čo málo povie. Napr:

Premennú, ktorá udáva uplynutý čas, pomenovať namiesto `int days;` radšej `int elapsedTimeInDays;`

3.2.5.2.9 Malé metódy a triedy

Metódy a triedy by mali byť malé, teda by nemali mať veľa riadkov. Čím menej tým lepšie, ale musí to byť funkčne použité.

Metóda by mala robiť jednu vec. Ak robí viac vecí tak sa dá rozbiť na viacero metód. Podobne aj trieda.

3.2.5.2.10 Angličtina

Je nutné v zdrojovom kóde VŽDY používať angličtinu!

3.2.6 Návod na používanie Git-u (pp 2.7)

3.2.6.1 Základ

Vytvorenie novej vetvy s menom branchname v lokálnom repozitári.

- *git branch branchname*

Pred samotnou prácou sa treba „prepnúť“ sa na danú vetvu vývoja.

- *git checkout branchname*

Pre prezeranie všetkých vetiev v lokálnom repozitári

- *git branch –a*

V príklade je uložené meno vzdialeného repozitára servera(origin) spolu s odkazom na server.

- *git remote add origin git@gitbus.fiit.stuba.sk:robocup-a55-kickers/robocup-a55-kickers.git*

Vzdialeným repozitárom môže byť aj iný repozitár na lokálnom počítači. Takýchto vzdialených repozitárov môžem v prípade potreby vytvoriť viac.

Vymazanie vetvy branchname v lokálnom repozitári

- *git branch -d branchname*

Pripadne, že som na vetve branchname a chcem ju vymazať treba prepnúť na inú vetvu.

Keď vetva obsahuje konflikty a chcem sa jej zbaviť, treba

- *git branch -D branchname*

Po zmene implementácii treba explicitne povedať, ktoré súbory podstúpili zmeny, prípadne pribudli ako nové v projekte a treba ich brať v úvahu pre nasledujúci commit.

- *git add .*

Zmeny nahrá do repozitára a spravíme ich viditeľnými v rámci lokálneho repozitára.

- *git commit -m "message"*

Pred zaslaním vetvy na vzdialený repozitár treba stiahnuť vetvu zo vzdialeného repozitára. Uvedený príklad stiahne vetvu branchname z adresy uloženej v origin a vetva je uložená s menom origin/branchname v lokálnom repozitári.

- *git fetch origin branchname*

Porovnanie súborov, odhalenie prípadných konfliktov medzi lokálnou vetvou a stiahnutou vetvou zo vzdialeného repozitára.

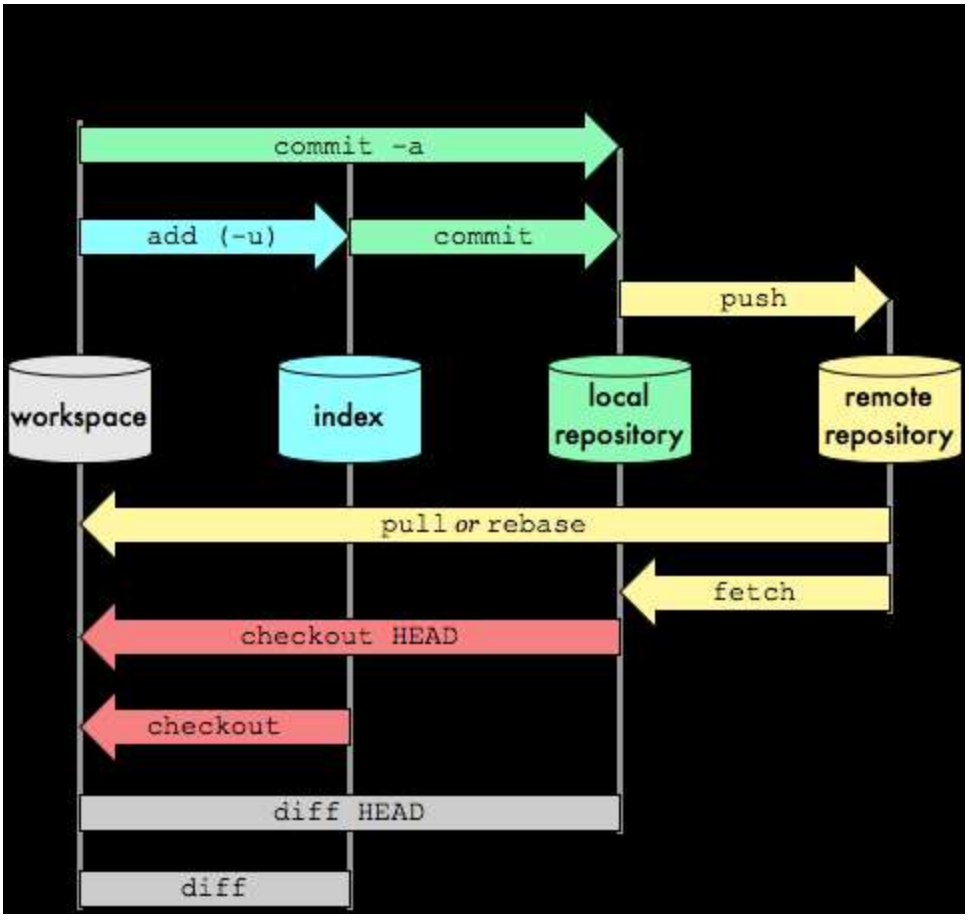
- *git diff branchname origin/branchname*

Zlúčenie vetiev, výsledok zlúčenia na vetve, na ktorej som nastavený.

- *git merge branchname origin/branchname*

Zaslanie vetvy branchname na vzdialený repozitár servera. V prípade, že vetva na vzdialenom repozitári neexistuje bude vytvorená ako nová.

- *git push origin branchname*



Obrázok 3.9: Architektúra komunikácie so vzdialeným repozitárom

Príkaz na globálne ignorovanie nechcených súborov v repozitári:
cd ~
vim .gitignore_global
git config --global core.excludesfile ~/.gitignore_global
do .gitignore_global napíšte :
*/nbproject/
build.xml
manifest.mf

3.2.6.2 Ďalšie príkazy

Fetch a merge vykonať naraz.

- *git pull origin branchname*

V tomto prípade keď vetvy obsahujú konflikty, vetva bude rozbitá. Ak chcem obnoviť stav vetvy, na ktorej som nastavený, na posledný commit.

- *git reset --hard*

Všetky zmeny od posledného commitu budú nenávratne vymazané.

Sledovanie zmien.

- git reflog

Nastavenie vetvy na vybraný minulý stav.

- git reset --hard HEAD~1